世界知的所有権機関
国際事務局

PCT

特許協力条約に基づいて公開された国際出願

(51) 国際特許分類6
G06F 9/45

A1

(11) 国際公開番号 WO99/30231

(43) 国際公開日 1999年6月17日(17.06.99)

(21) 国際出願番号 PCT/JP97/04542

(22) 国際出願日 1997年12月10日(10.12.97)

(71) 出願人（米国を除くすべての指定国について）
株式会社 日立製作所(HITACHI, LTD.)[JP/JP]
〒101 東京都千代田区神田駿河台四丁目6番地 Tokyo, (JP)
(72) 発明者；および
(75) 発明者／出願人（米国についてのみ）
本川敬子(MOTOKAWA, Keiko)[JP/JP]
〒216 神奈川県川崎市宮前区宮前平1丁目7番2号
Kanagawa, (JP)
西山博泰(NISHIYAMA, Hiroyasu)[JP/JP]
〒216 神奈川県川崎市宮前区神木本町1丁目2番3号
プラザ神木B-205 Kanagawa, (JP)
菊池純男(KIKUCHI, Sumio)[JP/JP]
〒195 東京都町田市三輪緑山3丁目22番8号 Tokyo, (JP)
(74) 代理人
弁理士 小川勝男(OGAWA, Katsuo)
〒100 東京都千代田区丸の内一丁目5番1号
株式会社 日立製作所内 Tokyo, (JP)

(81) 指定国 JP, US, 欧州特許 (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

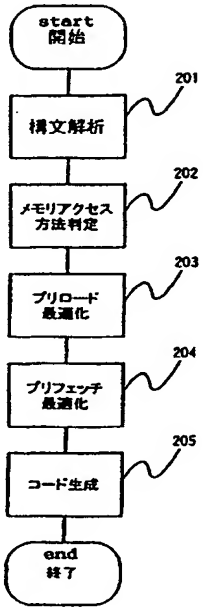
添付公開書類
国際調査報告書

(54)Title: MEMORY ACCESS OPTIMIZING METHOD

(54)発明の名称 メモリアクセス最適化方法

(57) Abstract

A memory access optimizing method which judges an access method suitable for each memory access and executes the preload optimization and prefetch optimization, according to the judgement result, using an architecture equipped with a prefetch mechanism to write the data on a main storage into a cache and a preload mechanism to write the data on the main storage into a register without writing it into the cache. The memory access method judging step analyzes whether there is a designation of memory access method by a user or not. Moreover, the memory access method judging step investigates whether the data are already in cache or not, whether the data compete with other data for cache or not, whether the data are to be referred to again later or not, and whether the data meet the restriction on register resource or not.



201 ... analysis of sentence construction
202 ... judgment of memory access method
203 ... preload optimization
204 ... prefetch optimization
205 ... code generation

(57)要約

主記憶上のデータをキャッシュに書き込むプリフェッチ機構と、主記憶上のデータをキャッシュに書き込まずにレジスタに書き込むプリロード機構とを備えたアーキテクチャを対象として、各メモリアクセスに適したアクセス方法を判定し、その判定結果に従ってプリロード最適化およびプリフェッチ最適化を実施するメモリアクセス最適化方法を提供する。メモリアクセス方法判定ステップは、ユーザによるメモリアクセス方法の指定があるかどうかを解析する。また、メモリアクセス方法判定ステップは、データが既にキャッシュにあるかどうか、他のデータとキャッシュ競合するか、データが後で再参照されるか、レジスタ資源の制約を満たすかを調べる。

明 細 書

メモリアクセス最適化方法

技術分野

本発明は、キャッシュ記憶をそなえたプロセッサをターゲットとするコンパイラに関し、特に配列要素参照のロード方式を最適化するメモリアクセス最適化方法に関する。

背景技術

マイクロプロセッサの速度向上に伴って主記憶アクセスのレイテンシが増大し、プログラムの実行性能に与える影響が増している。多くのプロセッサでは、主記憶よりもアクセスが速く比較的小容量のキャッシュ記憶を備え、レイテンシの大きい主記憶へのアクセス回数を削減している。すなわちメモリアクセス命令は、キャッシュヒット時には短いレイテンシでキャッシュをアクセスし、キャッシュミス時にのみ、主記憶をアクセスする。

キャッシュミス時のメモリアクセスのレイテンシを隠す1つの手法として、プリフェッチ（ソフトウェア・プリフェッチ）がある。プリフェッチ命令は、主記憶上のデータをノンブロッキングでキャッシュ上にロードする。コンパイラでは、ソフトウェアパイプライニング等のスケジューリング技法を用いて、事前にプリフェッチ命令を発行しておき、プリフェッチが完了するまでの間に別の演算を実行する。その後、ロード命令によりキャッシュ上のデータをアクセスする。この方法によりメモリアクセスによるレイテンシを隠すことができる。このようなプリフェッチの方法については、例えば「Todd C. Mowry 他:Design and Evaluation of Compiler Algorithm for Prefetching, Architectural Support for Programming Languages and Operating Systems, pp. 62-73, 1992」などに述べられている。

メモリアクセスのレイテンシを隠す別の機構として、プリロードと呼ばれる方法がある。これについては、「澤本他：RISC超並列スーパーコンピュータのデータバス技術、情報処理３８巻６号、pp.485-492、1997年」に述べられている。プリロード命令は、主記憶上のデータをキャッシュをバイパスして直接レジスタに書き込む。コンパイラでは、ソフトウェアパイプライニング等のスケジューリング技法を用いて、プリロード命令とデータを使用する演算命令をメモリレイテンシ以上離すことにより、メモリレイテンシを隠す。

プリロードは、プリフェッチと比較して次のような利点がある。プリロード１命令で主記憶からレジスタへのロードができるので、プリフェッチのように命令を付加する必要がなく、発行命令数が増加しない。キャッシュへの書き込みがないため、メモリスループットがよい。また、プリフェッチはデータの使用前にキャッシュから追い出されることがあるが、プリロードは直接レジスタに書き込むのでその心配はない。

一方、プリフェッチには次の利点がある。プリフェッチ命令はレジスタを占有しないため、レジスタプレッシャを増大しない。またプリフェッチ命令は、1度のメモリ要求で１キャッシュライン分をまとめてキャッシュに書き込み、連続データのアクセスではこれを有効利用できる。

前記の澤本らの文献にあげられているアーキテクチャは、プリフェッチとプリロードの両方の機構を備えており、プリロードは浮動小数点データに、プリフェッチは固定小数点と浮動小数点データの両方に適用できる。そこで固定小数点データはプリフェッチでキャッシュに、浮動小数点データはプリロードで直接レジスタにロードするコードを生成することが記載されている。１つのループ内の浮動小数点データに対して、プリフェッチとプリロードの２つの方式を使い分けることに関しては述べられていない。

プリフェッチに関しては、各メモリアクセスに対してプリフェッチが必要かどうかを解析し、冗長なプリフェッチ命令を削除する方法が研究されている。これは、前記の Mowry らによる文献などに述べられている。この方法はループ

ネストのリユース解析に基づく。同じキャッシュラインのデータを２度以上参照するとき、リユースがあるという。リユースは、１つの参照が異なるループイタレーション間で同じキャッシュラインをアクセスする自己リユースと、複数の参照間の群リユースに分類される。リユース解析では、配列の添字式をループ制御変数の線形式として表現し解析する。リユース解析の方法については、「M. E. Wolf and M. S. Lam: A Data Locality Optimizing Algorithm, Programming Language Design and Implementation, pp.30-44, 1991」に詳しく述べられている。冗長なプリフェッチ命令削除では、群リユースに着目する。群リユースを持つ複数の参照の中で、最初に新たなデータを参照するものをリーディング参照とする。リーディング参照にはプリフェッチを適用し、他のデータはこのプリフェッチでキャッシュに書き込んだデータを利用するのでプリフェッチを省略する。このようにして必要なデータだけにプリフェッチ命令を出す。

ループ間のリユース解析を実施して、プリフェッチを削減する最適化については、「Keith Cooper 他：Cross-loop Reuse Analysis and its Application to Cache Optimizations, Workshop on Languages and Compilers for Parallel Computing, pp.1-15, 1996」に述べられている。本文献によれば、各ループ内の配列の参照部分を求め、データフローで伝播することによりループ入口およびループ出口に到達するデータ集合を求める。ループ入口と出口の両方に到達するデータはプリフェッチ不要とすることで、プリフェッチを削減する。

上記で述べたように、プリフェッチとプリロードにはそれぞれの利点・欠点がある。従来のように、全てのデータをプリフェッチ、または全てのデータをプリロードする方法では、欠点がでてしまう。そこで、各メモリ参照の特性に応じて、プリフェッチまたはプリロードのうちより適した方法を選択し、プリフェッチとプリロードを併用するコードを生成することによって、両方の方式の利点を活用することができる。

本発明の目的は、プリロードとプリフェッチの２つの方式を適用可能なメモ

リ参照を対象に、その参照により適したアクセス方法を選択し、プリロードとプリフェッチを併用したコードを生成することによって、より実行性能の高いコードを生成する最適化方法を提供することにある。

発明の開示

本発明の目的は、各メモリ参照に対してプリフェッチ、プリロード、又はロードの、どのアクセス方法を選択するかを決定するメモリアクセス方法判定ステップと、プリロードと判定されたメモリアクセスを対象に最適化を実施しプリロードコードを生成するプリロード最適化ステップと、プリフェッチと判定されたメモリアクセスを対象にプリフェッチコードを生成するプリフェッチ最適化ステップによって達成される。

メモリアクセス方法判定ステップの第１の方法は、そのメモリアクセスに対してソースプログラムの記述またはコンパイラオプションによるメモリアクセス方法の指定があるかどうかを解析するステップと、その結果に応じてメモリアクセス方法を決定するステップとから成る。

メモリアクセス方法判定ステップの第２の方法は、データが既にキャッシュにあるかどうかを判定するステップと、他のデータとのキャッシュ競合を判定するステップと、データが後で再参照されるかどうかを判定するステップと、レジスタ資源の制約を満たすかどうかを判定するステップとから成り、データが既にキャッシュにあるかどうか、およびデータが後で再参照されるかを判定するステップは、それぞれループ内に関する解析とループ間に関する解析から成る。

図面の簡単な説明

第１図は、本発明に係るメモリアクセス最適化方法を用いたコンパイラが稼動するシステムの構成図を示し、第２図は、本発明に係るメモリアクセス最適化方法を用いたコンパイラの処理手順を示し、第３図は、メモリアクセス方法

の指示を付加されたソースプログラムの一例であり、第４図は、コンパイラオプションによるメモリアクセス方法指示の一例であり、第５図は、第３図のプログラムに対する中間語の一例であり、第６図は、ループ表の一例であり、第７図は、アクセス方法登録表の一例であり、第８図は、本発明に係るメモリアクセス最適化方法におけるメモリアクセス方法判定の処理手順を示し、第９図は、本発明に係るメモリアクセス方法判定において、あるメモリアクセスの指定文中の指定の有無を解析する処理手順を示し、第１０図は、本発明に係るメモリアクセス最適化方法におけるプリロード最適化の処理手順を示し、第１１図は、第５図の中間語の基本ブロックＢ３に対応するＤＡＧの例であり、第１２図は、本発明に係るプリロード最適化におけるＤＡＧエッジ上へのレイテンシ設定の処理手順を示し、第１３図は、第５図の中間語に対してプリロード最適化を適用した後の中間語の一例であり、第１４図は、本発明に係るメモリアクセス最適化方法におけるプリフェッチ最適化の処理手順を示し、第１５図は、第１３図の中間語に対してプリフェッチ最適化を適用した後の中間語の一例であり、第１６図は、本発明に係るメモリアクセス最適化方法におけるコンパイラの解析によるメモリアクセス方法判定の処理手順を示し、第１７図は、ループ表の一例であり、第１８図は、本発明に係るメモリアクセス方法判定における先行ループの参照データ解析の処理手順を示し、第１９図は、本発明に係るメモリアクセス方法判定における後続ループの参照データ解析の処理手順を示し、第２０図は、リユース登録表の一例であり、第２１図は、本発明に係るメモリアクセス方法判定におけるアクセス方法選択の処理手順を示し、第２２図は、本発明に係るアクセス方法選択において、データがキャッシュ上にあるかを判定する処理手順を示し、第２３図は、本発明に係るアクセス方法選択におけるキャッシュ競合解析の処理手順を示し、第２４図は、本発明に係るアクセス方法選択において、データが後で再参照されるかを判定する処理手順を示し、第２５図は、本発明に係るアクセス方法選択において、データが後で再参照されるかを判定する処理手順を示し、第２６図は、本発明に係るアクセス方法選

択におけるレジスタ資源制約の判定の処理手順を示し、第２７図は、ソースプログラムの一例であり、第２８図は、ソースプログラムの一例である。

発明を実施するための最良の形態

以下、図面を用いて本発明の実施の形態について説明する。

本発明の実施の形態の第１の例は、ユーザ指示に従ったメモリアクセス方法判定によるメモリアクセス最適化である。

第１図は、本発明のメモリアクセス最適化方法を適用したコンパイラが稼動する計算機システムの構成図である。この計算機システムは、ＣＰＵ１０１、ディスプレイ装置１０２、キーボード１０３、主記憶装置１０４、および外部記憶装置１０５より構成されている。キーボード１０３により、ユーザからのコンパイラ起動命令を受け付ける。コンパイラ終了メッセージやエラーメッセージは、ディスプレイ装置１０２に表示される。外部記憶装置１０５には、ソースプログラム１０６とオブジェクトプログラム１０７が格納される。主記憶装置１０４には、コンパイラ１０８、コンパイル過程で必要となる中間語１０９、ループ表１１０、およびアクセス方法登録表１１１が格納される。リユース登録表１１２は第２の実施例のためのもので、本実施例では使用しない。コンパイル処理はＣＰＵ１０１によって制御され実行される。

第２図に、第１図のシステムで稼動するコンパイラ１０８の処理手順を示す。

コンパイラの処理は、構文解析２０１、メモリアクセス方法判定２０２、プリロード最適化２０３、プリフェッチ最適化２０４、コード生成２０５の順で行う。

構文解析２０１では、ソースプログラム１０６を入力として構文解析、及びループ解析を行い、中間コード１０９とループ表１１０を出力する。構文解析処理及びループ解析処理に関しては、例えば「エイホ、セシィ、ウルマン著：コンパイラ（サイエンス社、１９９０年）」に記述されている。中間コード及びループ表については、後で説明する。

メモリアクセス方法判定２０２では、各メモリアクセスごとに適用すべきアクセス方法を判定し、その判定結果をアクセス方法登録表１１０に登録する。このメモリアクセス方法判定２０２は、本発明の特徴となる部分であるので、後に詳細に説明する。

プリロード最適化２０３では、アクセス方法判定２０２でプリロードと判定されたメモリアクセスを対象に、ソフトウェアパイプライニングの技法を用いた命令スケジューリング最適化を実施し、プリロードコードを生成する。プリフェッチ最適化２０４では、アクセス方法判定２０２でプリフェッチと判定されたメモリアクセスを対象に、プリフェッチコードを生成する最適化を実施する。プリロード最適化及びプリフェッチ最適化については、後で第１０図から第１５図を用いて説明する。

コード生成２０５では、中間コード１０９をオブジェクトプログラム１０７に変換し、出力する。

第３図に、アクセス方法指示が付加されたソースプログラムの例を示す。指示文３０１はプリフェッチ指示であり、プリフェッチすべき配列名または配列要素をカンマで区切って指定している。指示文は直後のループネスト内のメモリアクセスに対して有効である。指示文３０２は、プリロード指示である。

ユーザ指示は、ユーザがコンパイラ起動時にコンパイラオプションによって指定することもできる。第４図は、コンパイラオプションによるユーザ指定の例を示す。オプション４０１は、プリフェッチ指示オプションであり、指示文３０１と同様の意味を持つ。オプションには対象ループと、配列名または配列要素を指定できる。対象ループ「loop1」は、プログラム内の１番目のループネストを表す。オプション４０２はプリロード指示文である。

第５図は本実施例におけるコンパイラの中間コードの例で、第３図のソースプログラムに対応している。中間コードは構文解析２０１で作成する。中間コードは、基本ブロックをエッジで結んだグラフで表現されている。このようなグラフは制御フローグラフと呼ばれている。Ｂ０からＢ５は、それぞれ基本ブ

ロックを表す。基本ブロックは、複数の文で構成される。各文は実行命令にほぼ対応しているので、「命令」と呼ぶこともある。「t1=load(a[i][j])」は、a[i][j]の値をロードして一時変数 t1 に代入するという意味である。本実施例で最適化の対象となるメモリアクセスは「load」で表現されている。

第６図は本実施例におけるループ表の例である。このループ表は、構文解析２０１で作成される。６０１はループ番号、６０２はループに属する基本ブロックを示す。各ループにつけられたユーザ指示は、プリフェッチ指示文６０３、及びプリロード指示文６０４に登録される。

第６図のループ表は、第３図のソースプログラム及び第５図の中間コードに対応している。ループ番号１は内側ループを、ループ番号２は外側ループを表している。プリフェッチ指示文及びプリロード指示文は、内側ループであるループ番号１のループに対応するものとして、登録される。

第７図にアクセス方法登録表の一例を示す。アクセス方法登録表は、メモリアクセスが属するループを示すループ番号７０１、メモリアクセス７０２、アクセス方法７０３のフィールドを持つ。これらのフィールドは、メモリアクセス方法判定２０２で設定され、プリロード最適化２０３、プリフェッチ最適化２０４は、登録されたアクセス方法に従った処理を行う。

次に、アクセス方法判定２０２の処理手順を詳細に説明する。

第８図に、ユーザ指示によるアクセス方法判定の処理手順を示す。各ステップを説明する。

ステップ８０１は、中間コード上に未処理のメモリアクセスがあるかどうかを判定し、なければ処理を終了する。あれば、ステップ８０２へ進み、未処理のメモリアクセスを取り出し、REF とする。ステップ８０３では、REF の属するループを求め、LOOP とする。各参照が属するループは、中間コードからその参照が属する基本ブロックを求め、ループ表１１０で基本ブロックを探すことにより求められる。

ステップ８０４では、ループ表１１０を参照し、LOOP へのプリフェッチ指定

文が登録されているかどうかを調べる。あればステップ８０５へ進みそのプリフェッチ指定文を取り出し、ステップ８０６で指定文中に REF の指定があるかどうかを判定する。ステップ８０６の詳細については第９図を用いて後述する。指定がある場合にはステップ８０７へ進み、アクセス方法にプリフェッチを選択する。

プリフェッチの指定がない場合にはステップ８０８へ進み、LOOP へのプリロード指定があるかどうかを調べる。ある場合にはステップ８０９でプリロード指定文を取り出し、ステップ８１０で指定文中に REF の指定があるかどうかを調べる。ステップ８１０の詳細については第９図を用いて後述する。あればステップ８１１に進み、プリロードを選択する。

プリロード指定がない場合にはステップ８１２へ進み、ロードを選択する。

ステップ８１３では、選択したアクセス方法をアクセス方法登録表に登録する。

ステップ８０６、およびステップ８１０において、指定文中に REF の指定があるかどうかを調べる処理手順の詳細を示したものが第９図である。第９図の各ステップを説明する。

ステップ９０１は、指示文中に未処理の指定項目があるかどうかを判定する。ない場合はステップ９０７へ進み、REF の指定なしと判定して処理を終了する。ある場合にはステップ９０２へ進み、指定文から次の指定項目を取り出す。指定項目は配列名または配列参照を指定している。

ステップ９０３では配列要素の指定かどうか調べる。そうであればステップ９０４へ進み、配列要素が REF と一致するかどうか調べる。一致した場合はステップ９０６へ進み、REF の指定ありと判定して処理を終了する。一致しない場合には、ステップ９０１へ戻り、次の指定項目の処理へ進む。

ステップ９０３で配列要素指定でないと判定した場合にはステップ９０５へ進む。この場合は配列名の指定であるので、配列名が REF の配列と一致するかどうか調べる。一致した場合にはステップ９０６へ進み、REF の指定ありとして

処理を終了する。一致しない場合にはステップ９０１へ戻る。

以上で第１の実施例におけるメモリアクセス方法判定２０２の処理手順の説明を終了する。

次に、第８図および第９図の処理手順に従い、第３図のプログラムに対するアクセス方法判定処理を説明する。

ステップ８０１、ステップ８０２、ステップ８０３と進み、基本ブロックＢ３内のメモリアクセスa[i][j]と、ループ１を取り出す。ステップ８０４、ステップ８０５と進み、ループ１へのプリフェッチ指定「a, c[i][j]」を取り出す。ステップ８０６で、第９図に従い、指定文中にa[i][j]の指定があるかを判定する。

ステップ９０１、ステップ９０２で、プリフェッチ指定の最初の指定項目「a」を取り出す。これは配列名の指定である。ステップ９０３で配列要素指定ではないのでステップ９０５へ進む。ステップ９０５で、REF「a[i][j]」の配列名「a」と、指定項目「a」を比較し、一致するのでステップ９０６へ進み、指定ありとして終了する。

ステップ８０６で指定ありと判定されたので、ステップ８０７、ステップ８１３と進み、アクセス方法登録表に、「ループ１、a[i][j]、プリフェッチ」を登録する。

次に、ステップ８０１へ戻り、次のメモリアクセス「b[j][i]」について調べる。ステップ８０５でループ１のプリフェッチ指定「a, c[i][j]」を取り出し、b[j][i]の指定があるか調べる。指定がないので、ステップ８０８、ステップ８０９と進み、プリロード指定「b」を取り出す。ステップ８１０で指定があると判定する。ステップ８１１、ステップ８１３と進み、アクセス方法登録表に、「ループ１、b[j][i]、プリロード」を登録する。

次に、ステップ８０１へ戻り、次のメモリアクセス「c[i][j]」について調べる。ステップ８０５でループ１のプリフェッチ指定「a, c[i][j]」を取り出し、ステップ８０６で、第９図に従い判定する。

ステップ９０１、ステップ９０２で、最初の指定項目「a」を取り出す。配列名の指定であるのでステップ９０５へ進み、「c[i][j]」の配列名と一致しないので、ステップ９０１へ戻る。次に２番目の指定項目「c[i][j]」を取り出す。配列要素の指定であるのでステップ９０３からステップ９０４へと進む。配列要素が一致するので、ステップ９０６へ進み、指定ありとする。

ステップ８０６で指定ありと判定されたので、ステップ８０７、ステップ８１３と進み、アクセス方法登録表に、「ループ１、c[i][j]、プリフェッチ」を登録する。

次に、ステップ２０３のプリロード最適化について説明する。

プリロード最適化は、最内側ループを対象に、ループ単位に処理を行う。プリロード最適化は、基本的にはソフトウェアパイプライニングの技法に従った処理を行う。ソフトウェアパイプライニングについては、例えば、「M. Lam : Software Pipelining: An Effective Scheduling Technique for VLIW Machines, Proc. Of the SIGPLAN '88 Conference on Programming Language Design and Implementation, pp.318-328, 1988」に述べられている。第１０図は、１つのループに対するプリロード最適化の処理手順を示している。以下、第５図の中間コードに対する処理を例にあげながら、各ステップについて順に説明する。

ステップ１００１は、DAG(directed acyclic graph)を作成する。第１１図にDAGの例を示す。これは、第５図の中間コードの基本ブロックＢ３に対応している。DAGの各ノードは、基本ブロック中の各文に対応している。ノード間のエッジは、実行順序の制約を表す。例えば、ノード１１０３のmul演算は、ノード１１０１及びノード１１０２のloadの結果をオペランドとするので、ノード１１０１とノード１１０２は、ノード１１０３よりも先に実行しなければならない。この関係をノード１１０１からノード１１０３へのエッジ、及びノード１１０２からノード１１０３へのエッジで表す。

ステップ１００２は、ステップ１００１で作成したDAGのエッジ上にレイテンシを設定する。本ステップの処理手順の詳細を第１２図に示す。第１２図の

各ステップについて説明する。

本処理は、DAG 上のエッジを辿り、順に処理する。ステップ１２０１では未処理のエッジがあるかどうか判定し、なければ処理を終了する。あれば、ステップ１２０２へ進み、未処理のエッジを取り出して、処理対象とする。

ステップ１２０３は、エッジの始点ノードがメモリアクセスかどうか調べる。メモリアクセスでなければ、ステップ１２０７へ進み、始点ノードの演算に対応するレイテンシの値をエッジ上に設定して、このエッジの処理を終了し、ステップ１２０１へ戻る。ステップ１２０３でメモリアクセスであれば、ステップ１２０４へ進む。

ステップ１２０４では、始点ノードのメモリアクセスのアクセス方法を、アクセス方法登録表１１１により調べる。アクセス方法がプリロードであればステップ１２０５へ進み、主記憶アクセスのレイテンシを設定する。プリロードでなければステップ１２０６へ進み、キャッシュアクセスのレイテンシを設定する。

第５図の中間コードの例では、第１１図のようにレイテンシが設定される。演算 mul および add に対するレイテンシは 2 サイクルであるとする。また、メモリアクセス a[i][j]および c[i][j]は、アクセス方法がプリフェッチであるので、キャッシュアクセスのレイテンシを設定する。この値は 2 サイクルとする。また、メモリアクセス b[j][i]はアクセス方法がプリロードであるので、主記憶アクセスのレイテンシとして１０サイクルを設定している。

ステップ１００３では、設定したレイテンシの値をもとに、DAG ノードをソフトウェアパイプライニングのステージに分割する。本例では、DAG ノード１１０２を始点とするエッジのレイテンシが１０サイクルと大きいことから、ノード１１０２を第１ステージとし、その他のノードを第２ステージとする。

ステップ１００４でソフトウェアパイプライニングによる命令スケジューリングで、中間コードを変更する。

ステップ１００５で、プリロード参照の中間コードを「preload」に変更する。

第５図の中間コードに対してプリロード最適化を適用した結果を、第１３図に示す。

基本ブロックＢ２には第１ステージのDAGノード１１０２に対応する命令が、プロローグコードとして挿入される。基本ブロックＢ３がカーネルコードである。基本ブロックＢ４には、第２ステージに対応する命令がエピローグコードとして挿入されている。

配列ｂをアクセスする３つの命令は「preload」に変更されている。

この例では、スライドウィンドウ機構を利用したコードを示している。スライドウィンドウ機構に関しては、「澤本他：RISC超並列スーパーコンピュータのデータパス技術、情報処理３８巻６号pp.485-492、1997年」に述べられている。基本ブロックＢ３内では、２イタレーション後に使用するｂの要素をプリロードする。B3内の「slide(1)」は、イタレーション毎に、レジスタウィンドウを１ずつスライドする。したがって、b[i][j]をオペランドとする演算は、レジスタ frl を使用することとし、プリロード命令では２つ先のレジスタであるfr3に値を書き込んでいる。

以上でプリロード最適化２０３の説明を終了する。

次に、プリフェッチ最適化２０４について説明する。

プリフェッチ最適化の処理手順を第１４図に示す。各ステップについて説明する。

ステップ１４０１では何イタレーション先をプリフェッチするかを求め、その値をｎとする。この値の求めかたについては、例えば、「Todd C. Mowry他：Design and Evaluation of a Compiler Algorithm for Prefetching, ASPLOS V pp.62-73,1992」に記載されている。

ステップ１４０２では、ループ内のメモリ参照に対して、メモリアクセス方法登録表１１１によりアクセス方法がプリフェッチであるかどうかを調べ、未処理のプリフェッチ参照を取り出す。なければ終了する。

ステップ１４０３では、取り出したメモリアクセスに対して、ｎイタレーシ

ョン後にアクセスするアドレスのプリフェッチコードを作成する。ステップ１４０４では、ステップ１４０３で作成したプリフェッチコードをループ内に挿入し、ステップ１４０２へ戻る。

第１３図の中間コードに対してプリフェッチ最適化を適用した結果を第１５図に示す。ステップ１４０１では、ｎの値を２にしている。メモリアクセス「a[i][j]」はアクセス方法がプリフェッチであるから、ステップ１４０３において２イタレーション先のプリフェッチコードとして「prefetch a[i][j+2]」を作成する。ステップ１４０４でこのコードを中間コードに挿入する。第１５図ではa[i][j]のloadの直前に挿入している。メモリアクセス「c[i][j]」に対しても同様にプリフェッチコードを作成し、挿入する。

以上で、本発明の実施形態の第１の例の説明を終了する。

次に、本発明の実施形態の第２の例として、コンパイラの解析によるアクセス方法判定について述べる。

第１図に示したコンパイラが稼動するシステムの例に関しては、第１の実施例と同様である。

コンパイラの処理手順は、第１の実施例と同様に第２図に従う。

ステップ２０１では構文解析を行い、中間コード及びループ表を作成する。中間コードの表現は、第１の実施例と同様である。ループ表に関しては、第１７図を用いて後述する。リユース登録表１１２は、メモリアクセス方法判定２０２で使用する。これについては後で第２０図を用いて説明する。

ステップ２０２では、中間語およびループ表の情報を解析し、メモリアクセス方法を判定してアクセス方法登録表に登録する。アクセス方法登録表は、第１の実施例と同様で、第７図のようである。メモリアクセス方法判定の処理手順については、本実施例の特徴となる部分であるので、後で詳細に説明する。

ステップ２０３では、メモリアクセス方法登録表にプリロードと登録されたメモリアクセスを対象とした最適化を行う。ステップ２０４では、プリフェッチと登録されたメモリアクセスを対象とした最適化を行う。ステップ２０５で

は、これらの最適化を施した中間語を対象にコード生成を実施し、オブジェクトコードを生成する。これらのステップについては第１の実施例と同様であるので、説明を省略する。

以下、本実施例におけるメモリアクセス方法判定２０２に関して、詳細に説明する。

メモリアクセス方法判定は、ループ内の解析と、ループ間の解析を含んでいる。以下では、ループ内及びループ間の解析を両方実施する処理手順について説明するが、一方を省略して判定することも可能である。

第１６図は、本実施例におけるアクセス方法判定の処理手順を示す。以下、各ステップについて説明する。

ステップ１６０１では、ループの処理順序を決める。なるべく実行順にループを処理するように順序付ける。処理対象のループは最内側ループとする。順序付けた結果は、ループ表に登録される。本ステップは、ループ間の解析に関係する。ループ内の解析だけを実施する場合には、任意の順序でよい。

ループ表を第１７図に示す。１７０１はループ番号を示す。１７０２はループに属する基本ブロックを示す。各メモリアクセスが属するループを調べるには、属する基本ブロックを探せばよい。１７０３は前のループ番号、１７０４は次のループ番号を設定する。これらのフィールドにはループを順序付けた結果が設定される。これらのフィールドは、最内側ループに対して設定される。フィールド１７０５から１７０８はループ制御変数に関する情報である。これらはリユース解析などで配列添字を解析する際に利用される。

ステップ１６０２では、ステップ１６０１で決めたループの処理順序に従い、次の処理ループを取り出す。次のループがなければ処理を終了する。

ステップ１６０３はループ間の解析に関係し、先行ループの参照データを解析する。先行ループとは、現在の処理ループよりも前に実行されるループを指す。ループの処理順序が実行順になっていれば、先行ループは既にアクセス方法判定の処理済みであるので、先行ループ内の参照のアクセス方法を用いた解

析を行う。

本ステップの詳細な処理手順を示したものが、第１８図である。各ステップについて説明する。

ステップ１８０１では、前のループがあるかどうかを判定する。前のループはループ表の前ループフィールド１７０３に従って辿る。前のループがあればそれを取り出してステップ１８０２へ進み、なければ処理を終了する。

ステップ１８０２では、取り出した前のループ内に未処理のメモリアクセスがあるかどうかを調べる。あればステップ１８０３へ進み、なければステップ１８０６へ進む。ステップ１８０３では、メモリアクセスのアクセス方法を、アクセス方法登録表により調べ、プリフェッチまたはロードならばステップ１８０４へ進む。そうでない場合、即ちプリロードの場合には、ステップ１８０２へ戻り、次のメモリアクセスを処理する。

ステップ１８０４およびステップ１８０５は、プリフェッチまたはロード、即ちキャッシュをアクセスするメモリアクセスに関する処理である。ステップ１８０４では、本メモリアクセスのループでの参照範囲を求める。ステップ１８０５では、その参照範囲を先行ループの新参照データ集合に加える。例えば、メモリアクセスa[i]があり、ループ制御変数ｉの範囲が$1 \leqq i \leqq 100$であるとき、a[1:100]を新参照データ集合に加える。

ステップ１８０６では、新参照データ集合がキャッシュ容量以下かどうかを調べる。キャッシュ容量以下であればステップ１８０７へ進み、参照データ集合に新参照データ集合を設定した後、ステップ１８０１へ戻り、次の前ループを処理する。新参照データ集合がキャッシュ容量を越えていれば、処理を終了する。これは、前のループをこれ以上調べても、現在のアクセス方法判定の対象ループの入口では、キャッシュから追い出されてしまっているからである。この場合、最後に処理したループの参照データは、参照データ集合には加えられていない。

ステップ１６０４もループ間の解析に関係し、後続ループの参照データを解

析する。後続ループとは、現在の処理ループよりも後に実行されるループを指す。本ステップの詳細な処理手順を第１９図に沿って説明する。

ステップ１９０１では、ループ表の次ループフィールド１７０４に従い、次のループを取り出す。次のループがなければ処理を終了する。

ステップ１９０２では取り出したループ内に未処理のメモリアクセスがあるかを調べる。あればステップ１９０３へ進み、なければステップ１９０５へ進む。

ステップ１９０３では、メモリアクセスのループでの参照範囲を求め、ステップ１９０４ではその参照範囲を後続ループの新参照データ集合に加える。

ステップ１９０５では、新参照データ集合がキャッシュ容量以下かどうかを調べ、そうであれば、参照データ集合に新参照データ集合を設定した後、ステップ１９０１へ戻り、そうでなければ処理を終了する。

ステップ１６０５は、ループ内の解析に関係する。本ステップではループ内データのリユース解析を実施し、その結果をリユース登録表１１２に登録する。

リユース登録表を第２０図に示す。フィールド２００１には、リユース解析の対象としたメモリアクセスを登録する。リユース解析の結果は自己リユース、群リユースに分けて登録される。自己リユースとは、１つのメモリアクセスが何度も参照されるときに同じキャッシュラインをアクセスすることによるリユース、群リユースとは、複数のメモリアクセス間で同じキャッシュラインを参照することによるリユースである。フィールド２００２には、自己リユースの有無を登録する。フィールド２００３には群リユースの情報を登録する。群リユースを持つメモリアクセスを集合にまとめ、リユースをもつ参照間で最初に参照されるメモリアクセスを選んでリーディング参照とする。これらを、フィールド２００４、２００５、２００６から成る表に登録する。フィールド２００３には、そのメモリアクセスが属する群リユースの番号を登録する。リユース解析の方法については、公知の技術であるので、詳しい説明は省略する。

ステップ１６０６からステップ１６０９は、ループ内の各メモリアクセスに

対する処理である。ステップ１６０６ではループ内に未処理のメモリアクセスがあるかを判定し、なければステップ１６０２に戻り、次のループの処理を行う。未処理のメモリアクセスがあれば、ステップ１６０７へ進み、メモリアクセスを取り出して処理対象とする。ステップ１６０８では、取り出したメモリアクセスのアクセス方法を選択する。ステップ１６０９では、ステップ１６０８で選択したアクセス方法を、アクセス方法登録表１１１に設定し、ステップ１６０６に戻って次のメモリアクセスを処理する。

ステップ１６０８のアクセス方法選択処理の詳細を示しているのが第２１図である。この処理手順は、本実施例の最も特徴的な部分である。各ステップについて順に説明する。

ステップ２１０１、２１０２、および２１０３は、キャッシュに関連した判定を行う。ステップ２１０１では、メモリアクセスの対象データがすでにキャッシュ上にあるかどうかを判定する。あればステップ２１０７へ進みロードを選択する。なければステップ２１０２へ進む。ステップ２１０２では、処理対象のメモリアクセスが、他のキャッシュデータとキャッシュ競合するかどうかを判定する。競合する場合にはステップ２１０６へ進み、プリロードを選択する。競合しない場合にはステップ２１０３へ進む。ステップ２１０３では、処理対象のメモリアクセスと同一キャッシュラインのデータを後で再び参照するかどうかを調べる。そうであればステップ２１０５へ進み、プリフェッチを選択する。そう出なければステップ２１０４へ進む。

ステップ２１０４はレジスタ資源に関する判定を行う。プリロードを適用するメモリアクセスは長時間にわたってレジスタを占有するため、プリロードの過剰使用によるレジスタ不足を防ぐことが目的である。レジスタ資源の制約条件を満たせばステップ２１０６へ進み、プリロードを選択する。満たさなければ、ステップ２１０５へ進み、プリフェッチを選択する。

ステップ２１０１、２１０２、２１０３、および２１０４の各条件判定の詳細については、以下で説明する。

ステップ２１０１の処理手順の詳細を示しているのが、第２２図である。各ステップについて説明する。

ステップ２２０１では、ステップ１６０３で解析した先行ループの参照データ集合を参照し、現在処理中のメモリアクセスが集合内に含まれているかどうかを判定する。含まれている場合には、先行ループでキャッシュに書き込まれているので、ステップ２２０４へ進み、キャッシュ上にありと判定して処理を終了する。先行ループの参照データ集合に含まれていない場合には、ステップ２２０２へ進む。

ステップ２２０２では、ステップ１６０５の解析により作成したリユース登録表を参照し、群リユースがあるかどうかを判定する。ない場合にはステップ２２０５へ進み、キャッシュ上にないと判定する。群リユースがあればステップ２２０３へ進み、群リユースのリーディング参照であるかどうかを判定する。群リユースは、リーディング参照の参照時に主記憶からキャッシュへの書き込みを行い、残りの参照はキャッシュを再利用する。従って、リーディング参照であるときは、ステップ２２０５へ進みキャッシュ上にないと判定する。そうでないときは、ステップ２２０４へ進み、キャッシュ上にあると判定する。

ステップ２２０１はループ間の解析に関するもので、ループ入口でデータがキャッシュにのっているかどうかを調べている。ステップ２２０２および２２０３は、ループ内の解析に関するもので、ループ開始からアクセス直前までの間に、他のメモリアクセスにより参照データがキャッシュに書き込まれるかどうかを調べている。

第２３図は、ステップ２１０２のキャッシュ競合判定の処理手順を示している。本解析はループ内の他の参照との間のキャッシュ競合を調べる。

ステップ２３０１では、アクセス方法登録表に同一ループに属する参照が登録されているかを調べ、あれば取り出してステップ２３０２へ進む。未処理の参照がなければステップ２３０６へ進み、キャッシュ競合しないと判定して処理を終了する。

ステップ２３０２では、取り出した参照のアクセス方法がプリフェッチまたはロードであるかを調べる。そうであればキャッシュに書き込まれるのでステップ２３０３へ進み、キャッシュ競合を調べる。そうでなければアクセス方法はプリロードであり、キャッシュへ書き込まれないので、ステップ２３０１へ戻る。

ステップ２３０３では、現在処理中の参照と、ステップ２３０１で取り出した参照の間のアドレス差を計算する。ステップ２３０４では、このアドレス差をキャッシュサイズで割った剰余が一定値以下であるかどうかを判定する。一定値以下であればステップ２３０５へ進み、キャッシュ競合すると判定して終了する。一定値以下でなければこの２つの参照はキャッシュ競合しないので、ステップ２３０１へ戻り、次の参照を調べる。

次に、ステップ２１０３の処理の詳細を説明する。

初めに、簡易的な判定方法として、各メモリアクセスの添字の解析による判定について述べる。第２４図に処理手順を示す。

ステップ２４０１では、配列添字を解析してストライドを計算する。ストライドは連続するイタレーションの参照間が配列要素いくつ分であるかを示す。配列添字がループ制御変数の１次式で表現されるときにストライドは一定で、ループ制御変数の増分値と添字式におけるループ制御変数の係数との積がストライド値である。

ステップ２４０２では、ストライドが定数であるかどうかを判定する。定数でなければステップ２４０５へ進み、再参照しないと判定して処理を終了する。定数であればステップ２４０３へ進む。

ステップ２４０３では、ストライド判定の基準値を計算する。１つのキャッシュラインをｎ回アクセスするためのストライド値として、キャッシュラインサイズを配列の要素サイズとｎの積で割った値を計算し、これをｓとする。

ステップ２４０３ではストライドの値が基準値ｓ以下であるかどうかを判定する。ｓ以下であればキャッシュラインをｎ回以上アクセスするので、ステッ

プ２４０５へ進み再参照すると判定して終了する。そうでなければステップ２４０６へ進み、再参照しないと判定する。

次に、ステップ２１０３の別の処理手順として、ステップ１６０４および１６０５の解析結果を用いる方法について説明する。第２５図に処理手順を示す。

ステップ２５０１では、リユース登録表を参照し、自己リユースがあるかどうかを判定する。あればステップ２５０６へ進み、再参照すると判定する。なければステップ２５０２へ進む。

ステップ２５０２では、リユース登録表を参照し、群リユースがあるかどうかを判定する。ある場合にはステップ２５０３へ進み、リーディング参照であるかどうかを判定する。リーディング参照である場合には同じ群の他の参照によるリユースがあるので、ステップ２５０６へ進み、再参照すると判定する。そうでない場合にはステップ２５０４へ進む。

ステップ２５０４はループ間の再参照を調べるステップである。ステップ１６０４で解析した後続ループの参照データを解析し、処理対象のメモリアクセスが含まれるかどうかを判定する。含まれればステップ２５０６へ進み、再参照すると判定する。含まれなければステップ２５０５へ進み、再参照しないと判定する。

以上で、第２４図、第２５図、およびステップ２１０３の処理手順の説明を終了する。

次に、ステップ２１０４のレジスタ資源の制約の判定について説明する。本ステップの処理手順の詳細を第２６図に示す。

ステップ２６０１はアクセス方法登録表を参照し、同じループのメモリアクセスで既にプリロードと決定したものの数を数える。

ステップ２６０２は、プリロード数がレジスタ資源の制約を満たすかどうかを判定するステップである。最適なレジスタ資源の制約条件はスケジューリングの方法などに依存するが、ここではレジスタ数に一定の係数ｋを掛けた値をプリロード数の上限とすることにする。例えば、レジスタ数を32本、係数を0.5

とする場合、プリロード数が16を越えないようにする。（プリロード数）＋1≦（レジスタ数）＊ｋであれば、ステップ２６０３へ進み、レジスタ資源の制約を満たすと判定する。そうでなければステップ２６０４へ進み、レジスタ資源の制約を満たさないと判定する。

以上で第２の実施例の処理手順の説明を全て終了する。

次にプログラムの例を用いて、本実施例による処理過程を説明する。

初めに単一ループのプログラムを例に説明する。第２７図はソースプログラムの例である。

第１６図の処理手順に従って、本プログラムのアクセス方法判定を実施する。

ステップ１６０１、１６０２、１６０３、１６０４はループ間の処理に関するものであるので、処理は不要である。

ステップ１６０５では、リユース解析を行い、第２０図に示した結果が得られる。メモリアクセス「c[x[i]]」は、ストライドが一定でないランダムは参照であるので、自己リユースを持たない。その他のメモリアクセスは、連続参照またはストライド２の参照であり、連続するイタレーション間で同じキャッシュラインをアクセスするので、自己リユースを持つ。群リユースは同じ配列の参照に対して解析される。配列「a」は、２つの参照「a[i]」「a[i+2]」を持つ。これらは同じイタレーション内で同じキャッシュラインをアクセスする可能性があるので、これらを群番号１として登録する。リーディング参照は「a[i]」とする。配列「b」も２つの参照を持つが、２つの参照間の距離が大きいため群リユースはなしと判定される。配列「d」の２つの参照は群リユースを持つ。「d[i+100]」でアクセスした要素を、100イタレーション後に「d[i]」がアクセスする。従って、「d[i+100]」をリーディング参照として、この群リユースを群番号２として登録する。

次に、各メモリアクセスに対するアクセス方法の判定（ステップ１６０８）について説明する。初めに「a[i]」のアクセス方法を判定する。ステップ２１０１では第２２図に従った判定を実施する。先行ループはなく、群リユースの

リーディング参照であることから、キャッシュ上になしと判定される。次にステップ２１０２へ進み、第２３図に従ってキャッシュ競合の解析を行う。ステップ２３０１で、アクセス方法登録表には他の参照は未登録であるので、ステップ２３０６へ進みキャッシュ競合しないと判定する。次にステップ２１０３で同一キャッシュラインを後で再参照するかを判定する。第２４図については省略し、第２５図の処理手順に従うものとする。ステップ２５０１で自己リユースがあるので、ステップ２５０６へ進み、再参照すると判定する。従って第２１図の処理ではステップ２１０５へ進み、プリフェッチを選択する。次に「a[i+2]」のアクセス方法を判定する。ステップ２１０１では、第２２図の処理手順に従い、群リユースがあってリーディング参照でないことから、ステップ２２０４でキャッシュ上にありと判定する。従ってステップ２１０７へ進みロードを選択する。次に配列ｂについて説明する。「b[2*i]」は自己リユースがあることからプリフェッチを選択する。「b[2*i+1024]」のステップ２１０２の処理について説明する。ステップ２３０１で、アクセス方法登録表にプリフェッチと登録された参照「b[2*i]」を取り出す。２データ間のアドレスは、8byte×1024=8Kbyte である。キャッシュサイズを 4Kbyte とすると、ステップ２３０４の判定により、２つのデータはキャッシュ競合すると判定される。２つともプリフェッチを適用してしまうと、一方をプリフェッチした後、他方のプリフェッチによりキャッシュから追い出され、プリフェッチの効果がなくなってしまう。このような場合には２つ目の参照はプリフェッチを選択せず、ステップ２１０６へ進んでプリロードを選択する。他の参照についての説明は省略する。このようにしてメモリアクセス方法を判定した結果は、a[i]、b[i*2]、x[i]、および d[i+100]がプリフェッチ、a[i+2]および d[i]がロード、c[x[i]]および b[i*2+1024]がプリロードとなる。

次に複数のループを含むプログラムの例を説明する。第２８図にソースプログラムを示す。

ステップ１６０１で、ループの処理順序を決める。処理順序は実行順序に従

うことが望ましいので、LOOP1, LOOP2, LOOP3, LOOP4 の順に処理を行う。ループ表は、第１７図のように登録されている。以下、「LOOP2」とコメント付けされた２番目のループの処理手順を説明する。

ステップ１６０３で先行ループの参照データを第１８図の処理手順に従って解析する。ステップ１８０１で前のループとして LOOP1 を取り出す。ステップ１８０２でメモリアクセス「a[i]」を取り出す。LOOP1 の「a[i]」のアクセス方法はプリフェッチが指定されているものとする。ステップ１８０４で参照範囲を求める。制御変数 i は下限値が 0、上限値が 99、増分値が 1 であるので、参照範囲は a[0:99]となる。ステップ１８０５で新参照データ集合は{a[0:99]}となり、ステップ１８０２に戻って他のメモリアクセスがないのでステップ１８０６へ進む。新参照データ集合で参照するデータの容量は、8byte×100=800byte である。本例では、キャッシュサイズを 4Kbyte と仮定する。新参照データ集合はキャッシュ容量以下であるので、ステップ１８０７で参照データ集合を{a[0:99]}とする。ステップ１８０１へ戻り、LOOP1 の前ループがないので処理を終了する。

次に、ステップ１６０４で後続ループの参照データを解析する。ステップ１９０１で次のループとして LOOP3 を取り出す。メモリアクセス「b[i]」を解析し、新参照データ集合は{b[0:399]}となる。この集合は 8byte×400=3200byte 参照し、キャッシュ容量以下であるので、これを参照データ集合とし、次に LOOP4 を調べる。メモリアクセス「c[i]」の参照範囲を新参照データ集合に加えると、{b[0:399], c[0:399]}となる。この集合は 6400byte を参照し、キャッシュサイズの 4Kbyte を越えているので、処理を終了する。この結果、後続ループの参照データ集合として{b[0:399]}が得られる。

次に、ステップ１６０５でループ内のメモリアクセス a[i], b[i*4], c[i*4] についてリユース解析を行う。群リユースは存在しない。a[i]は自己リユースを持つ。キャッシュラインサイズを 32byte とすると、b および c は 1 ラインを 1 回しかアクセスしないので、自己リユースをもたない。

次に各メモリアクセスのアクセス方法を選択する。最初にa[i]について調べる。ステップ２１０１でキャッシュ上にあるかを調べる。 ステップ２２０１で、a[i]の参照範囲a[0:99]は先行ループの参照データ集合に含まれるので、キャッシュ上にあると判定され、アクセス方法としてロードを選択する。次に、b[i*4]について調べる。ステップ２１０１で、bは先行ループの参照データ集合に含まれておらず、群リユースも持たないので、キャッシュ上にないと判定される。ステップ２１０２ではキャッシュ競合はないとする。ステップ２１０３で同一キャッシュラインデータを後で再参照するかを第２５図に従い調べる。リユースを持たないので、ステップ２５０１およびステップ２５０２の判定を経てステップ２５０４へ進む。b[i*4]の参照範囲はb[0:396]で後続ループの参照データ集合に含まれるので、再参照すると判定する。ステップ２１０５へ進み、プリフェッチを選択する。次に、c[i*4]について調べる。cは先行ループの参照データ集合に含まれておらず、群リユースを持たないので、キャッシュ上にはないと判定される。キャッシュ競合もないとする。第２５図の処理では、リユースを持たず、後続ループの参照データにも含まれないことから再参照しないと判定する。ステップ２１０４へ進み、レジスタ資源の制約を調べる。レジスタ数を３２、係数を0.5とすると、ループ内にプリロードと登録済みの他のメモリアクセスはないので制約を満たし、ステップ２１０６でプリロードを選択する。

以上で第２８図のプログラムに対するメモリアクセス方法判定の処理手順の説明を終了し、第2の実施例の説明を終える。

産業上の利用可能性

本発明によれば、ループ内の各メモリアクセスに対して、プリフェッチ、プリロード、またはロードなどの、そのメモリアクセスにより適したアクセス方法を選択することができる。その結果、プリロードとプリフェッチを併用したコードを生成することができ、より実行性能の高いオブジェクトコードを生成できる。

請 求 の 範 囲

1．主記憶上のデータをキャッシュ上に書き込むプリフェッチ機構と、主記憶上のデータをキャッシュ上には書き込まずにレジスタに直接ロードするプリロード機構とを備えたアーキテクチャにおいて、

プリフェッチとプリロード両方の機構を利用したコードを出力するメモリアクセス最適化方法であって、

プログラム上の各メモリアクセスに対して、キャッシュから直接レジスタへ書き込むロード、またはキャッシュへの書き込み後にキャッシュからロードするプリフェッチ、またはプリロードのどれを適用すべきかを判定するアクセス方法判定ステップと、

前記アクセス方法判定ステップでプリロードと判定された参照に対してプリロードコードを生成するステップと、

前記アクセス方法判定ステップでプリフェッチと判定された参照に対してプリフェッチコードを生成するステップとを有し、

前記アクセス方法判定ステップは、

前記プログラム中のアクセス方法の指定を解析するステップを有し、

前記解析によりプリフェッチまたはプリロードと指定された参照に対しては、前記指定に従ってアクセス方法を判定することを特徴とするメモリアクセス最適化方法。

2．前記アクセス方法の指定は、

プログラムソース内またはコンパイラコマンドラインに、ループ単位で指定し、プリフェッチまたはプリロードすべき配列名または配列要素を参照として指定することを特徴とする請求の範囲第1項記載のメモリアクセス最適化方法。

3．主記憶上のデータをキャッシュ上に書き込むプリフェッチ機構と、主記憶上のデータをキャッシュ上には書き込まずにレジスタに直接ロードするプリロード機構とを備えたアーキテクチャにおいて、

プリフェッチとプリロード両方の機構を利用したコードを出力するメモリアクセス最適化方法であって、

プログラム上の各メモリアクセスに対して、ロード、またはプリフェッチ、またはプリロードのどれを適用すべきかを判定するアクセス方法判定ステップと、

アクセス方法判定ステップでプリロードと判定された参照に対してプリロードコードを生成するステップと、

アクセス方法判定ステップでプリフェッチと判定された参照に対してプリフェッチコードを生成するステップとを有し、

前記アクセス方法判定ステップは、

キャッシュまたはレジスタ資源の利用状況に基づいた判定を実施することを特徴とするメモリアクセス最適化方法。

4．前記アクセス方法判定ステップは、

連続するイタレーション間でのメモリアクセスの参照距離が何要素分かというストライドを判定するステップと、

アクセス方法判定のためのストライド値の基準である基準ストライド値を決定するステップとを有し、

ストライドが基準ストライド値以下の参照に対しては、プリフェッチを選択し、その他の参照にはプリロードを選択して判定することを特徴とする請求の範囲第3項記載のメモリアクセス最適化方法。

5．前記基準ストライド値を決定するステップは、

1回のプリフェッチでキャッシュに書き込まれるデータサイズと、1回のメモリアクセスが参照するデータサイズとに基づき決定することを特徴とする請求の範囲第4項記載のメモリアクセス最適化方法。

6．前記アクセス方法判定ステップは、

メモリアクセス間での同一キャッシュラインの再参照を解析するステップを有し、

前に実行されるデータとの間に再参照があればロードを選択し、

後に実行されるデータとの間に再参照があればプリフェッチを選択し、

再参照がなければプリロードを選択して判定することを特徴とする請求の範囲第３項記載のメモリアクセス最適化方法。

７．前記アクセス方法判定ステップは、

メモリアクセス間でのキャッシュ競合を解析するステップを有し、

アクセス方法としてプリフェッチを選択した他のデータとキャッシュ競合を起こすデータに対しては、プリロードを選択して判定することを特徴とする請求の範囲第３項記載のメモリアクセス最適化方法。

８．前記アクセス方法判定ステップは、

アクセス方法にプリロードを選択したループ内のメモリアクセスに対してレジスタの占有状況を判定するステップを有し、

レジスタの占有状況が所定の条件を満たしている場合には、プリフェッチを選択して判定することを特徴とする請求の範囲第３項記載のメモリアクセス最適化方法。

９．前記メモリアクセス間での同一キャッシュラインの再参照を解析するステップは、

同じループに属する参照間での再参照を解析するステップと、

異なるループ間の参照間での再参照を解析するステップとから成り、

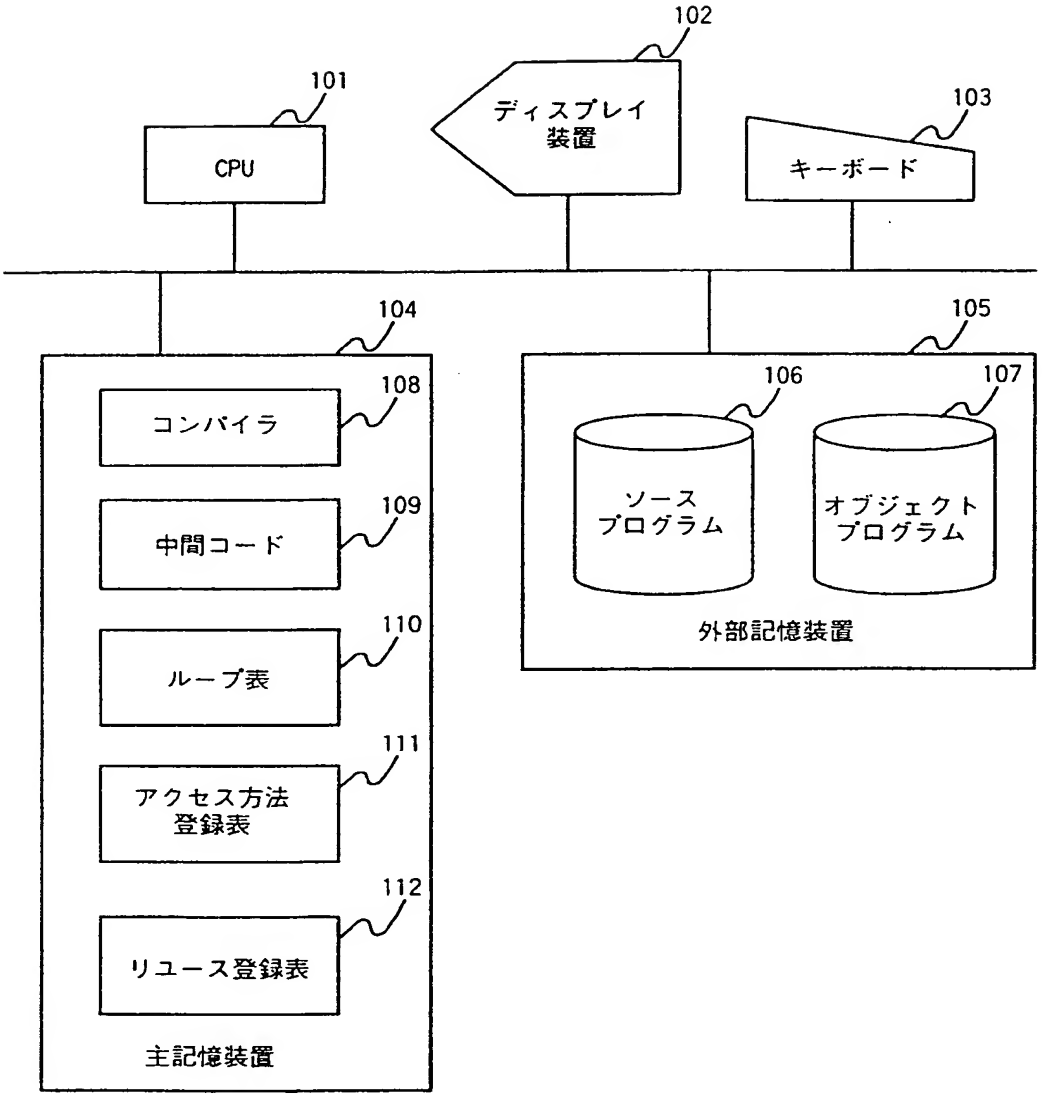
上記異なるループ間の再参照を解析するステップは、さらに、

前に実行されるループ内のプリフェッチまたはロードを選択したメモリアクセスの参照範囲の和を解析するステップと、

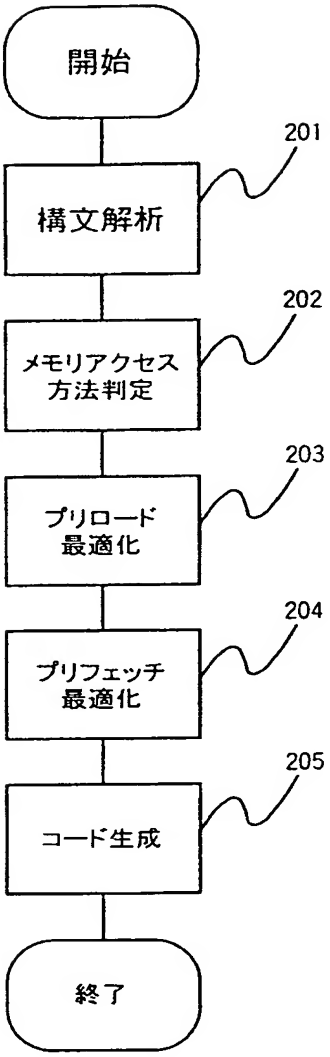
後に実行されるループ内のメモリアクセスの参照範囲の和を解析するステップを有する

ことを特徴とする請求の範囲第６項記載のメモリアクセス最適化方法。

第1図


102
101
ディスプレイ
装置
103
CPU
キーボード
104
105
108
106
107
コンパイラ
ソース
プログラム
オブジェクト
プログラム
109
中間コード
外部記憶装置
110
ループ表
111
アクセス方法
登録表
112
リユース登録表
主記憶装置

第２図


開始
201
構文解析
202
メモリアクセス
方法判定
203
プリロード
最適化
204
プリフェッチ
最適化
205
コード生成
終了

第3図

```
double a[100][100],b[100][100],c[100][100],s;
void func()
{
    int     i, j;
#prefetch   a, c[i][j]
#preload    b
    for (i = 0; i < 100; i++) {
        for (j = 0; j < 100; j++) {
            s = s + a[i][j] * b[j][i] + c[i][j];
        }
    }
}
```

301

302

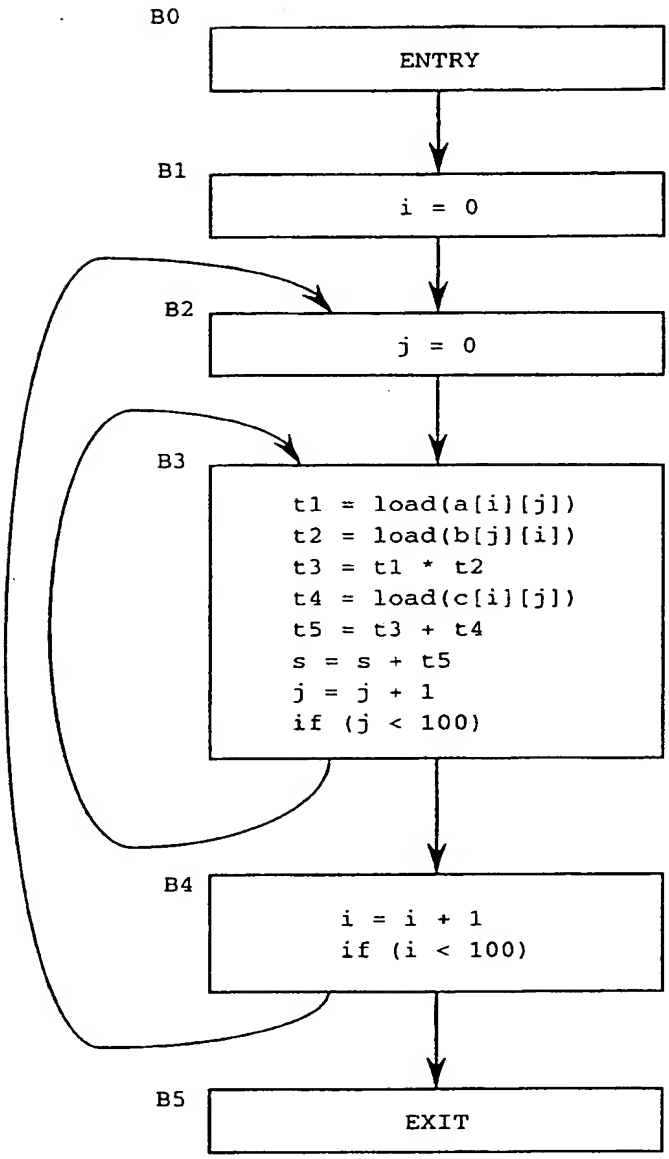
第4図

```
cc  -Oprefetch,loop1,a,c[i][j]  -Opreload,loop1,b  func.c
```

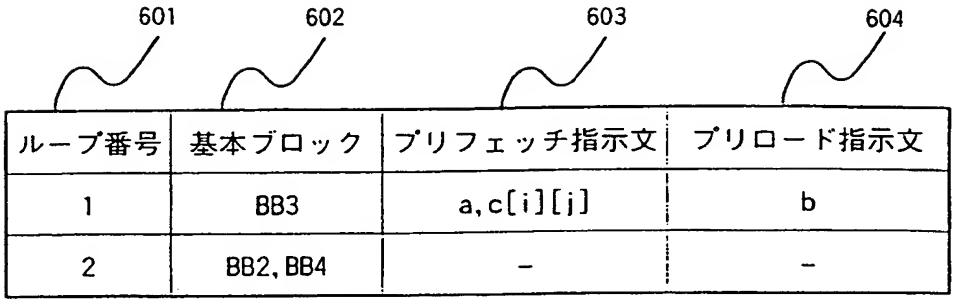
401

402

第5図

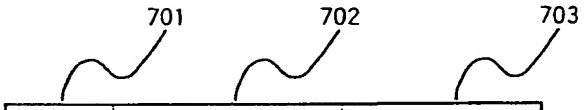

B0
ENTRY
B1
i = 0
B2
j = 0
B3
t1 = load(a[i][j])
t2 = load(b[j][i])
t3 = t1 * t2
t4 = load(c[i][j])
t5 = t3 + t4
s = s + t5
j = j + 1
if (j < 100)
B4
i = i + 1
if (i < 100)
B5
EXIT

第6図



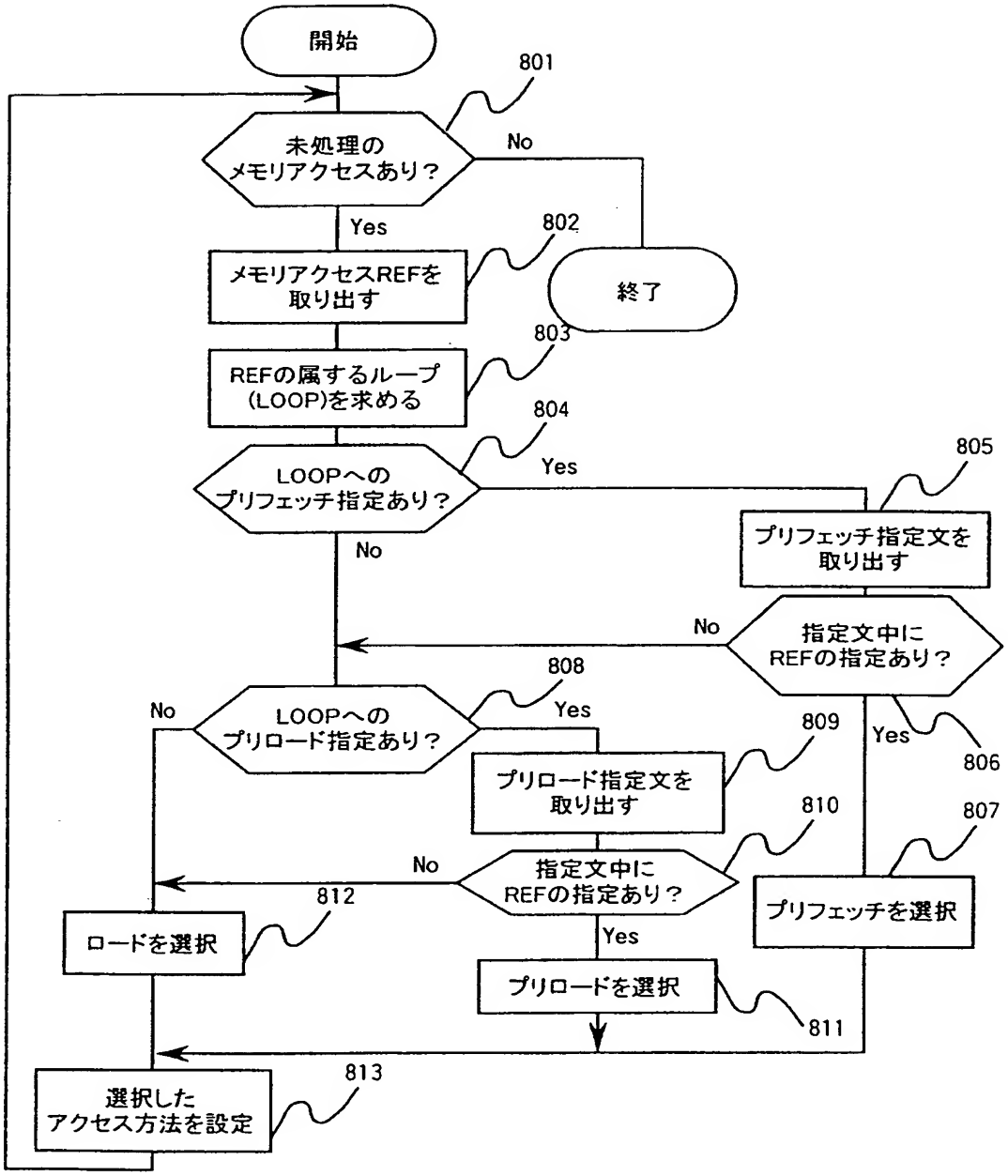
ループ番号	基本ブロック	プリフェッチ指示文	プリロード指示文
1	BB3	a, c[i][j]	b
2	BB2, BB4	-	-

第7図

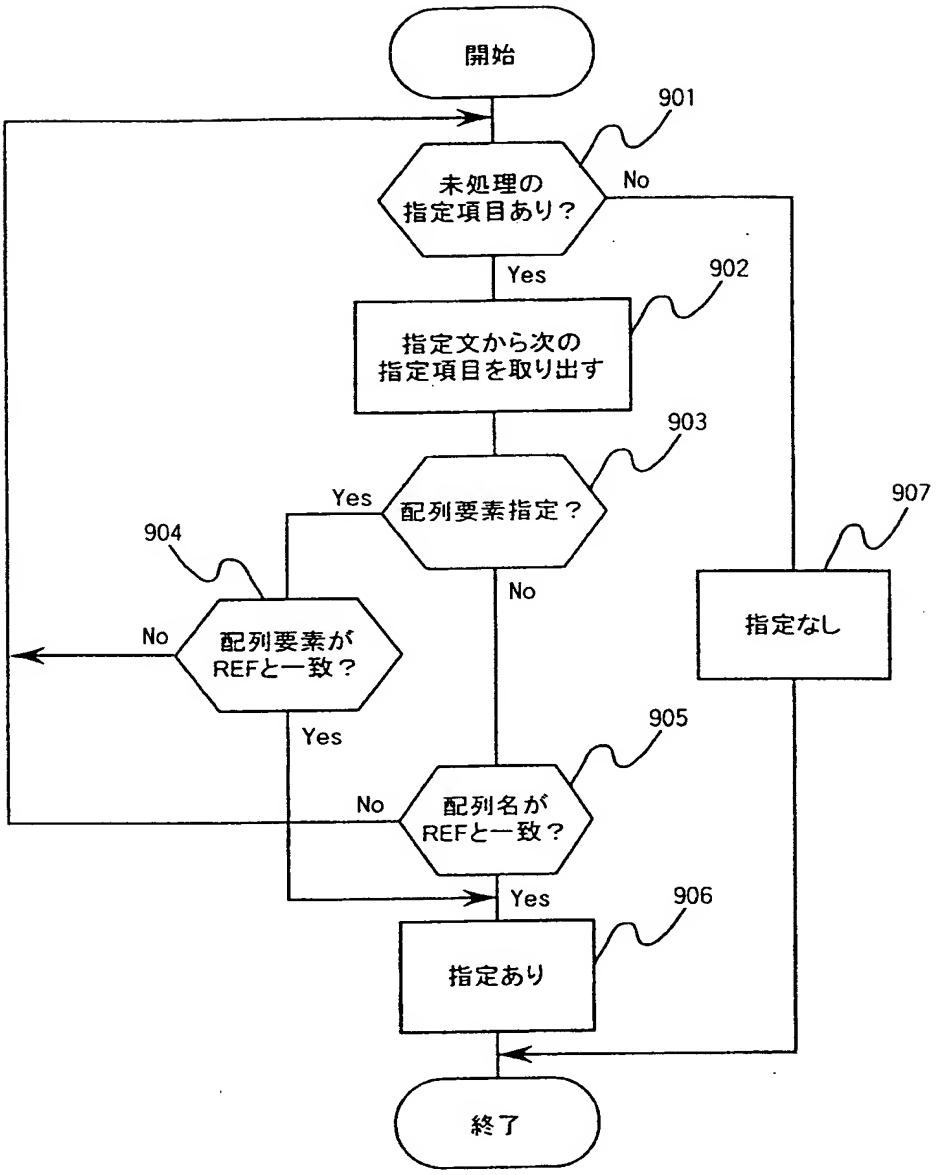


ループ	メモリアクセス	アクセス方法
1	a[i]	プリフェッチ
1	a[j]	プリロード
1	a[i+1]	ロード
2	b[j]	プリロード
3	c[k]	プリロード

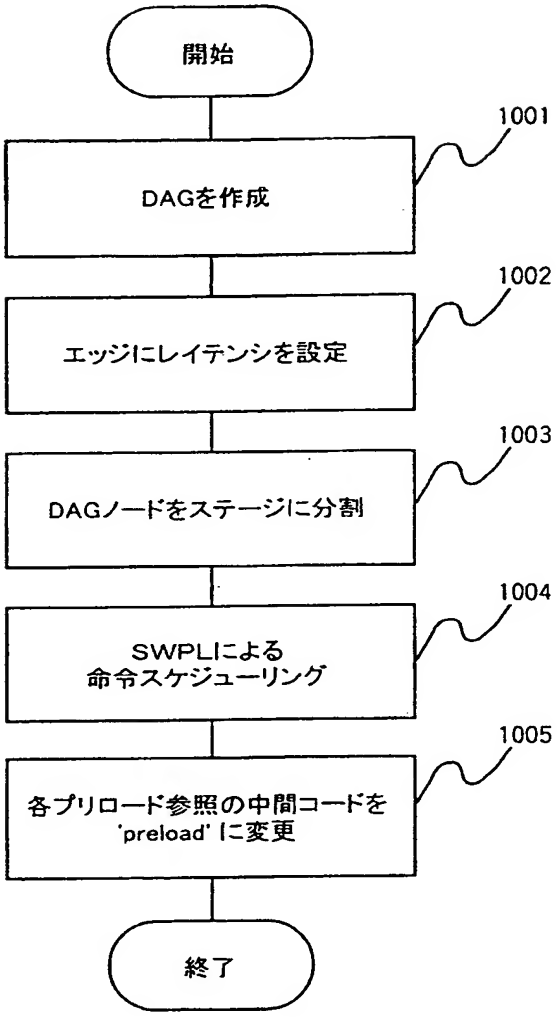
第8図


開始
801
未処理の
メモリアクセスあり？
No
Yes
802
メモリアクセスREFを
取り出す
終了
803
REFの属するループ
(LOOP)を求める
804
LOOPへの
プリフェッチ指定あり？
Yes
805
プリフェッチ指定文を
取り出す
No
No
指定文中に
REFの指定あり？
808
No
LOOPへの
プリロード指定あり？
Yes
809
プリロード指定文を
取り出す
Yes
806
810
807
No
指定文中に
REFの指定あり？
プリフェッチを選択
812
ロードを選択
Yes
プリロードを選択
811
813
選択した
アクセス方法を設定

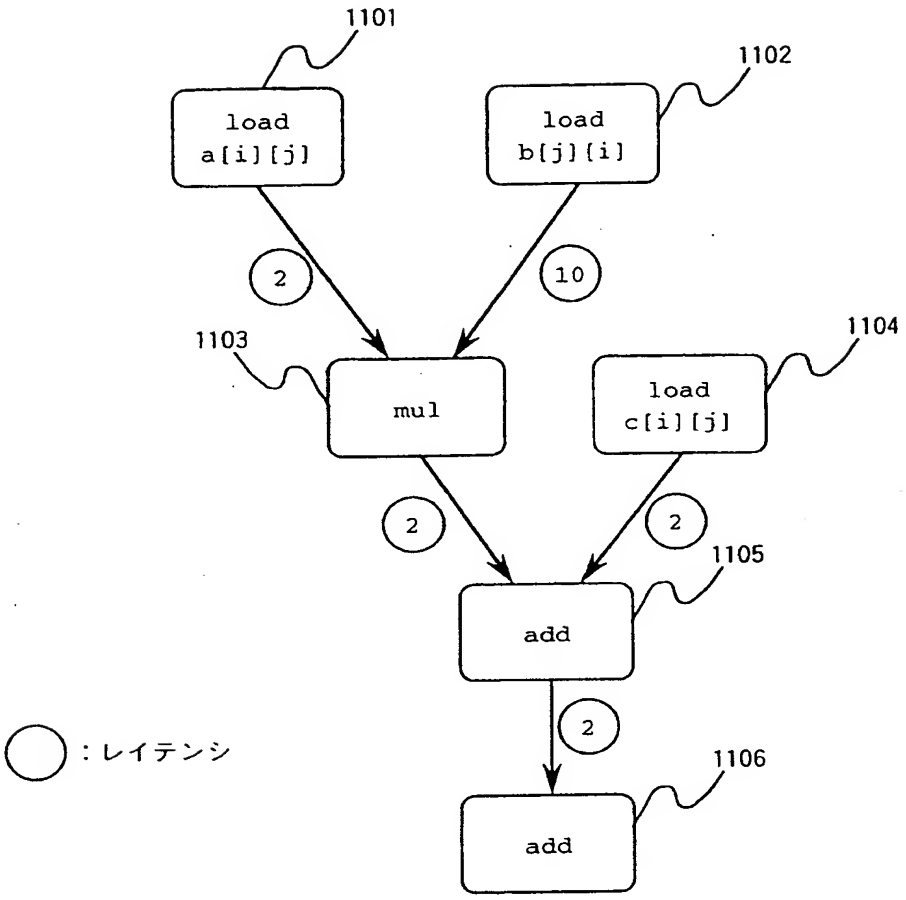
第9図


開始
901
未処理の
指定項目あり？
No
Yes
902
指定文から次の
指定項目を取り出す
903
配列要素指定？
Yes
907
904
No
配列要素が
REFと一致？
No
指定なし
Yes
905
No
配列名が
REFと一致？
Yes
906
指定あり
終了

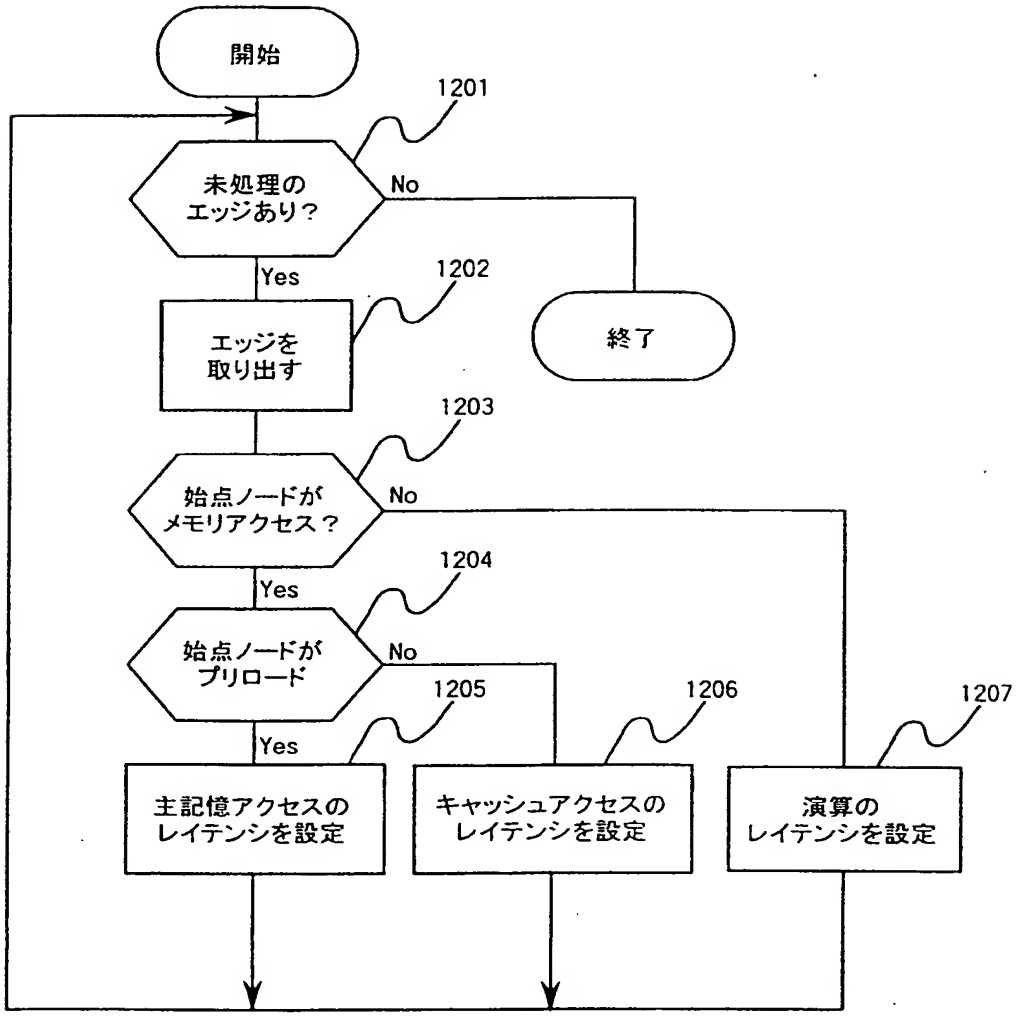
第１０図


開始
1001
DAGを作成
1002
エッジにレイテンシを設定
1003
DAGノードをステージに分割
1004
SWPLによる
命令スケジューリング
1005
各プリロード参照の中間コードを
'preload' に変更
終了

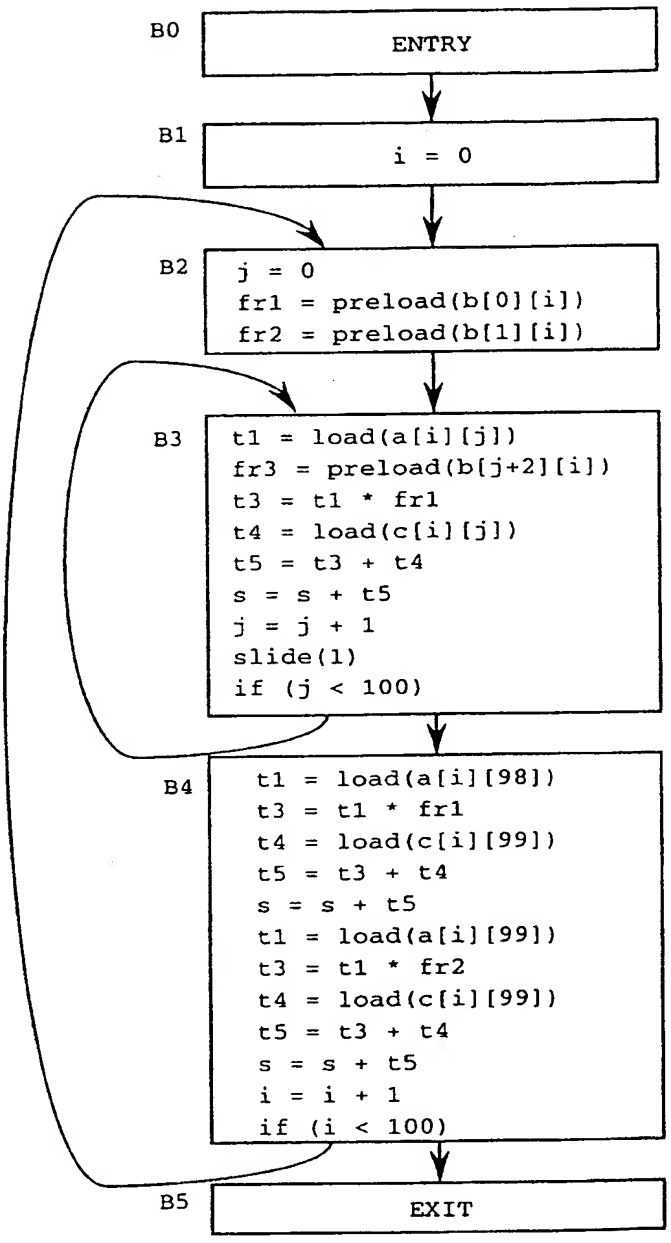
第１１図


1101
load
a[i][j]
1102
load
b[j][i]
2
10
1103
mul
1104
load
c[i][j]
2
2
1105
add
2
1106
add
：レイテンシ

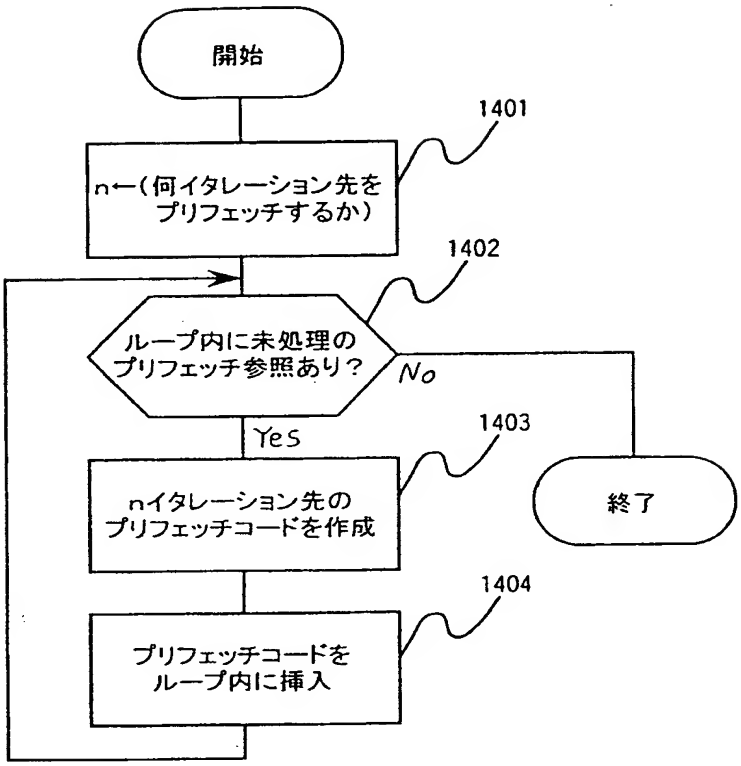
第１２図


開始
1201
未処理の
エッジあり？
No
Yes
1202
エッジを
取り出す
終了
1203
始点ノードが
メモリアクセス？
No
Yes
1204
始点ノードが
プリロード
No
Yes
1205
1206
1207
主記憶アクセスの
レイテンシを設定
キャッシュアクセスの
レイテンシを設定
演算の
レイテンシを設定

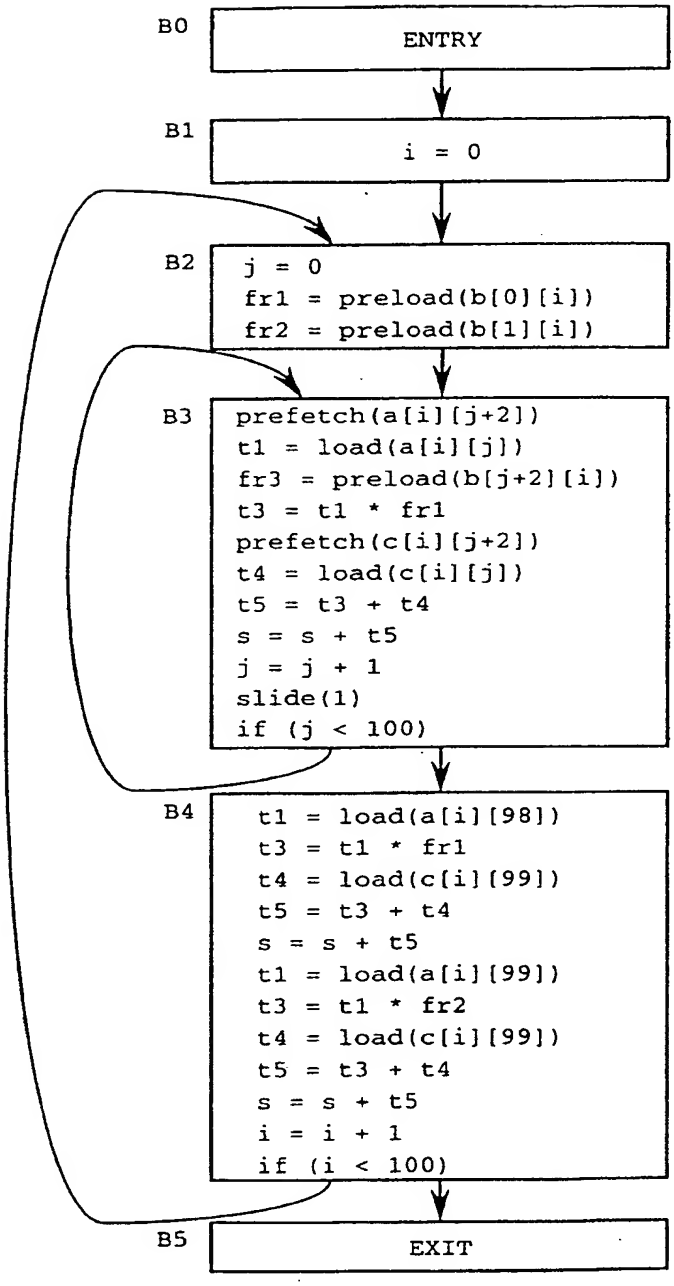
第13図


B0
ENTRY
B1
i = 0
B2
j = 0
fr1 = preload(b[0][i])
fr2 = preload(b[1][i])
B3
t1 = load(a[i][j])
fr3 = preload(b[j+2][i])
t3 = t1 * fr1
t4 = load(c[i][j])
t5 = t3 + t4
s = s + t5
j = j + 1
slide(1)
if (j < 100)
B4
t1 = load(a[i][98])
t3 = t1 * fr1
t4 = load(c[i][99])
t5 = t3 + t4
s = s + t5
t1 = load(a[i][99])
t3 = t1 * fr2
t4 = load(c[i][99])
t5 = t3 + t4
s = s + t5
i = i + 1
if (i < 100)
B5
EXIT

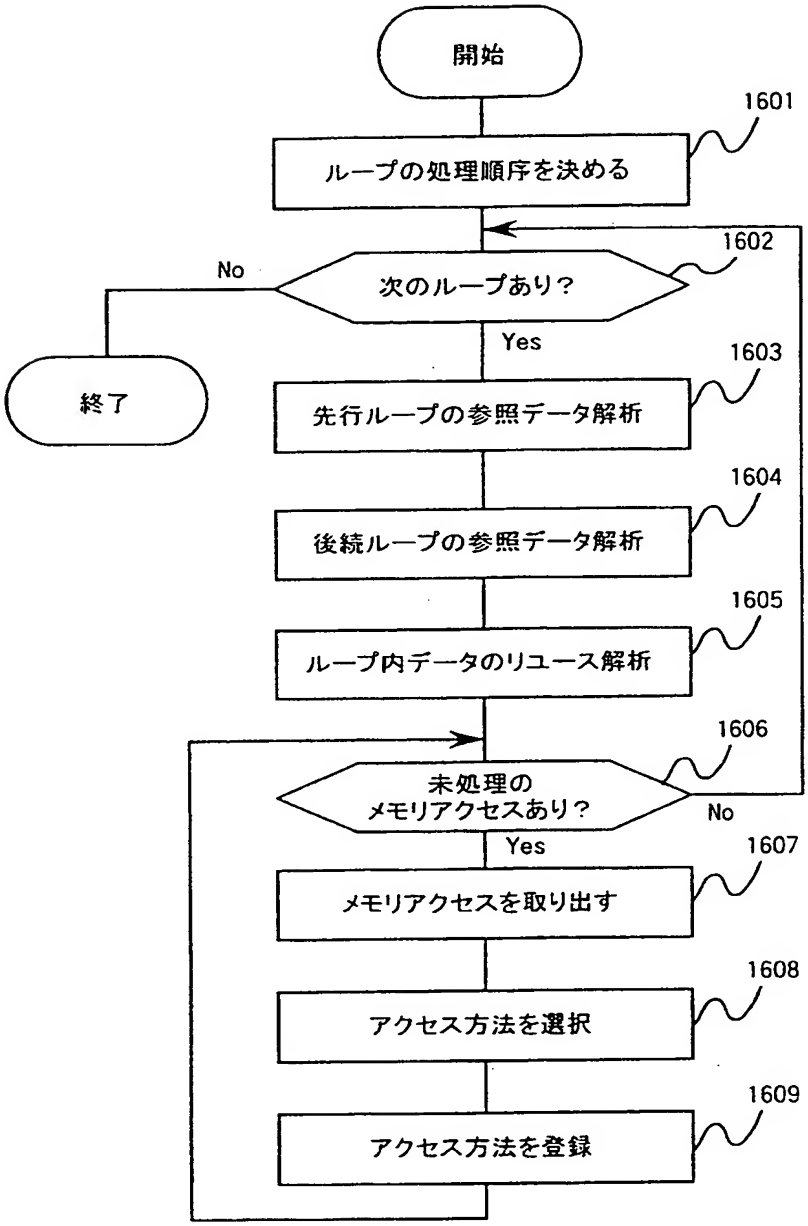
第１４図


開始
1401
n←(何イタレーション先を
プリフェッチするか)
1402
ループ内に未処理の
プリフェッチ参照あり?
No
Yes
1403
nイタレーション先の
プリフェッチコードを作成
終了
1404
プリフェッチコードを
ループ内に挿入

第15図


B0
ENTRY
B1
i = 0
B2
j = 0
fr1 = preload(b[0][i])
fr2 = preload(b[1][i])
B3
prefetch(a[i][j+2])
t1 = load(a[i][j])
fr3 = preload(b[j+2][i])
t3 = t1 * fr1
prefetch(c[i][j+2])
t4 = load(c[i][j])
t5 = t3 + t4
s = s + t5
j = j + 1
slide(1)
if (j < 100)
B4
t1 = load(a[i][98])
t3 = t1 * fr1
t4 = load(c[i][99])
t5 = t3 + t4
s = s + t5
t1 = load(a[i][99])
t3 = t1 * fr2
t4 = load(c[i][99])
t5 = t3 + t4
s = s + t5
i = i + 1
if (i < 100)
B5
EXIT

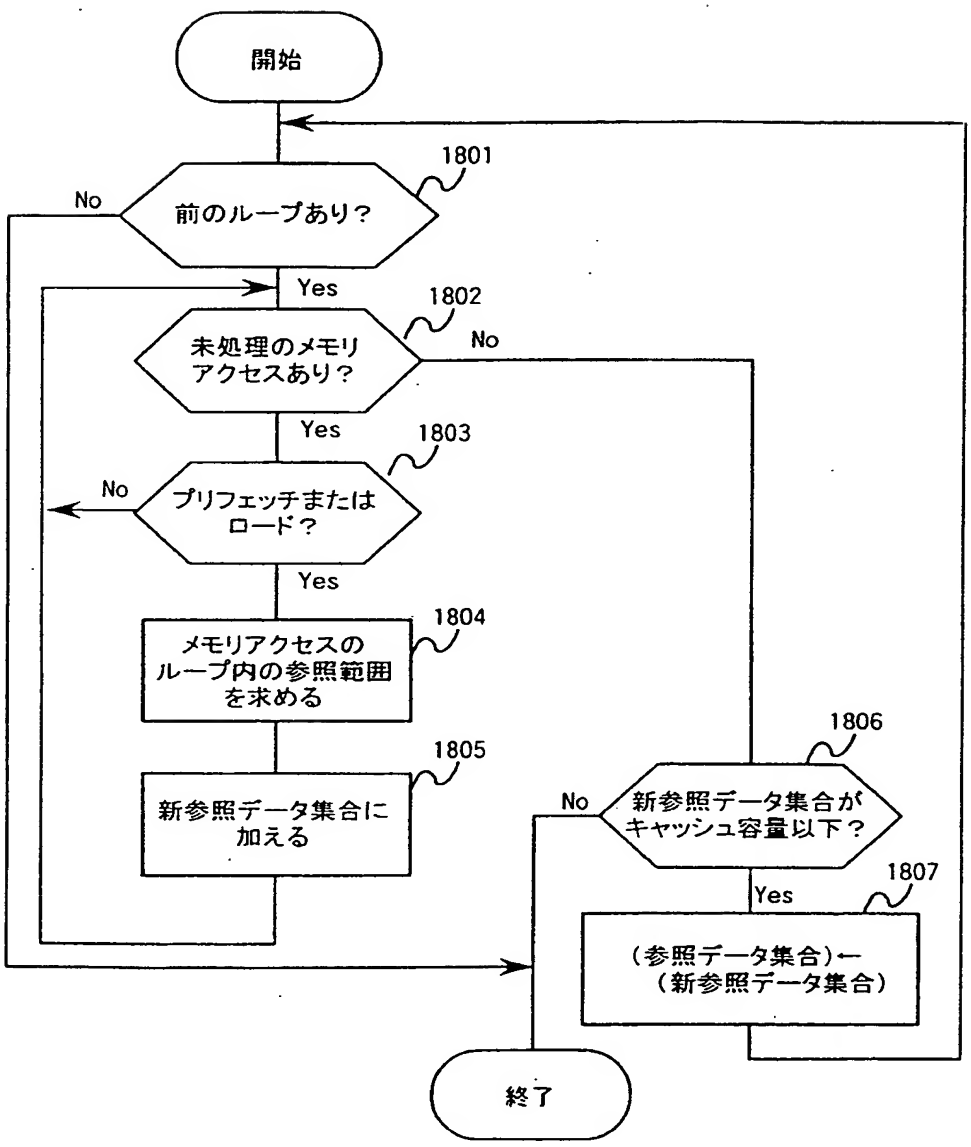
第16図


開始
1601
ループの処理順序を決める
1602
次のループあり？
No
Yes
終了
1603
先行ループの参照データ解析
1604
後続ループの参照データ解析
1605
ループ内データのリユース解析
1606
未処理の
メモリアクセスあり？
No
Yes
1607
メモリアクセスを取り出す
1608
アクセス方法を選択
1609
アクセス方法を登録

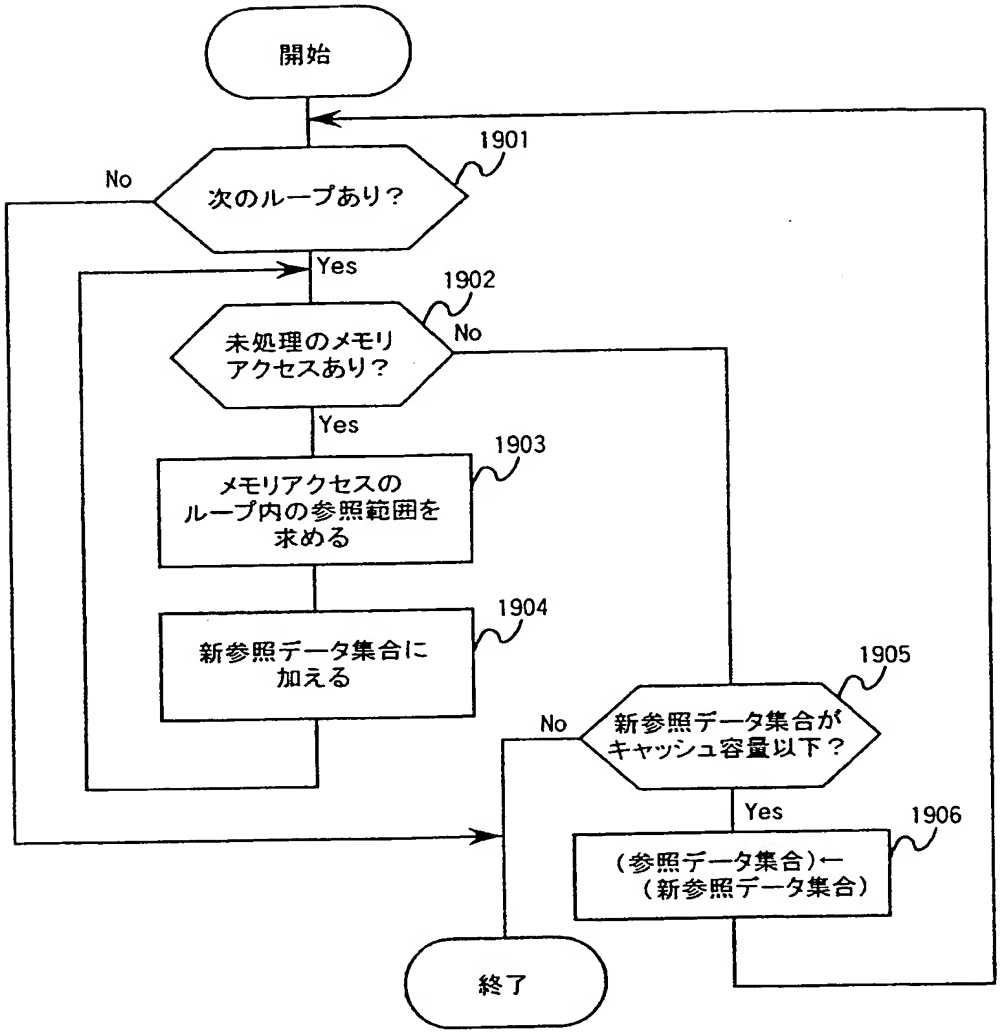
第１７図

1701	ループ番号	1	2	3	4
1702	基本ブロック	BB2	BB5	BB8	BB11
1703	前ループ	–	1	2	3
1704	次ループ	2	3	4	–
1705	制御変数	i	i	i	i
1706	制御変数下限値	0	0	0	0
1707	制御変数上限値	99	99	399	399
1708	制御変数増分値	1	1	1	1

第18図


開始
1801
前のループあり？
No
Yes
1802
未処理のメモリアクセスあり？
No
Yes
1803
プリフェッチまたはロード？
No
Yes
1804
メモリアクセスのループ内の参照範囲を求める
1805
新参照データ集合に加える
1806
新参照データ集合がキャッシュ容量以下？
No
Yes
1807
（参照データ集合）←（新参照データ集合）
終了

第19図


開始
1901
次のループあり？
No
Yes
1902
未処理のメモリ
アクセスあり？
No
Yes
1903
メモリアクセスの
ループ内の参照範囲を
求める
1904
新参照データ集合に
加える
1905
新参照データ集合が
キャッシュ容量以下？
No
Yes
1906
（参照データ集合）←
（新参照データ集合）
終了

第２０図

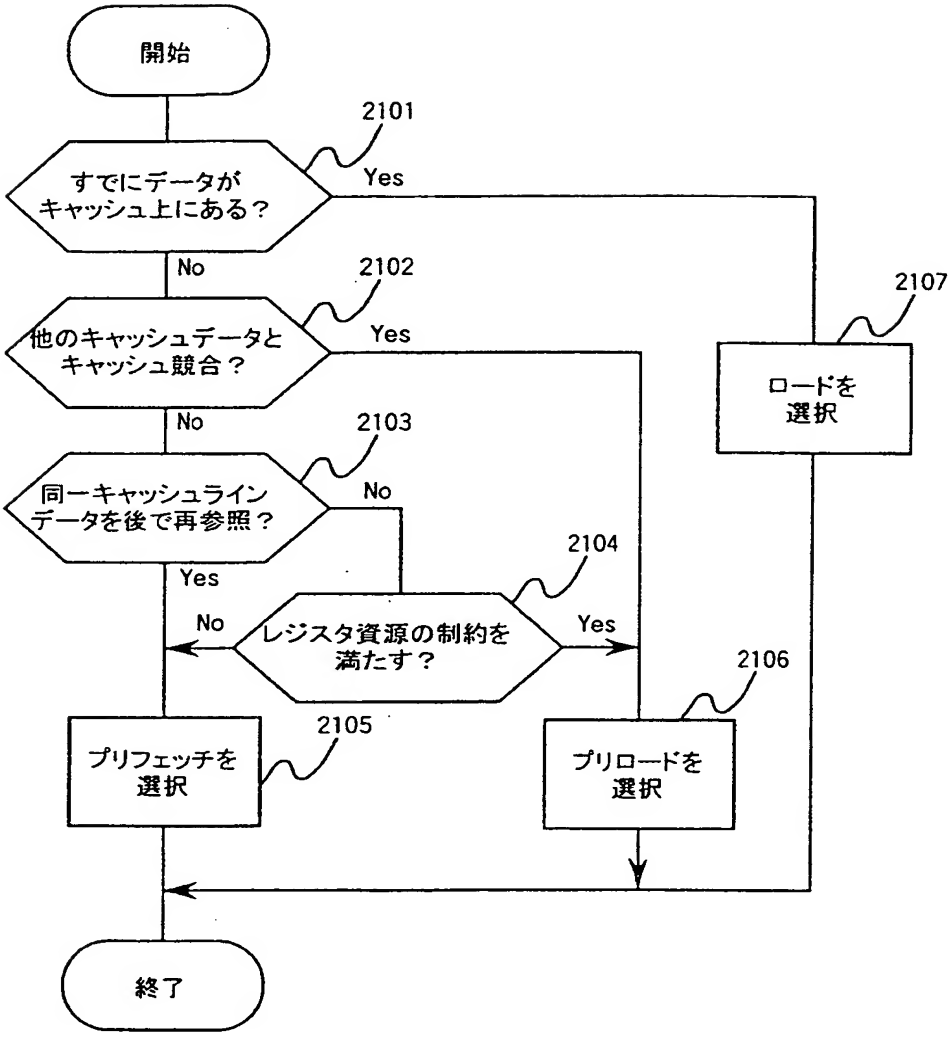
2001 2002 2003

メモリアクセス	自己リユース	群リユース
a[i]	あり	1
a[i+2]	あり	1
b[2*i]	あり	なし
x[i]	あり	なし
c[x[i]]	なし	なし
d[i]	あり	2
d[i+100]	あり	2
b[2*i+1024]	あり	なし

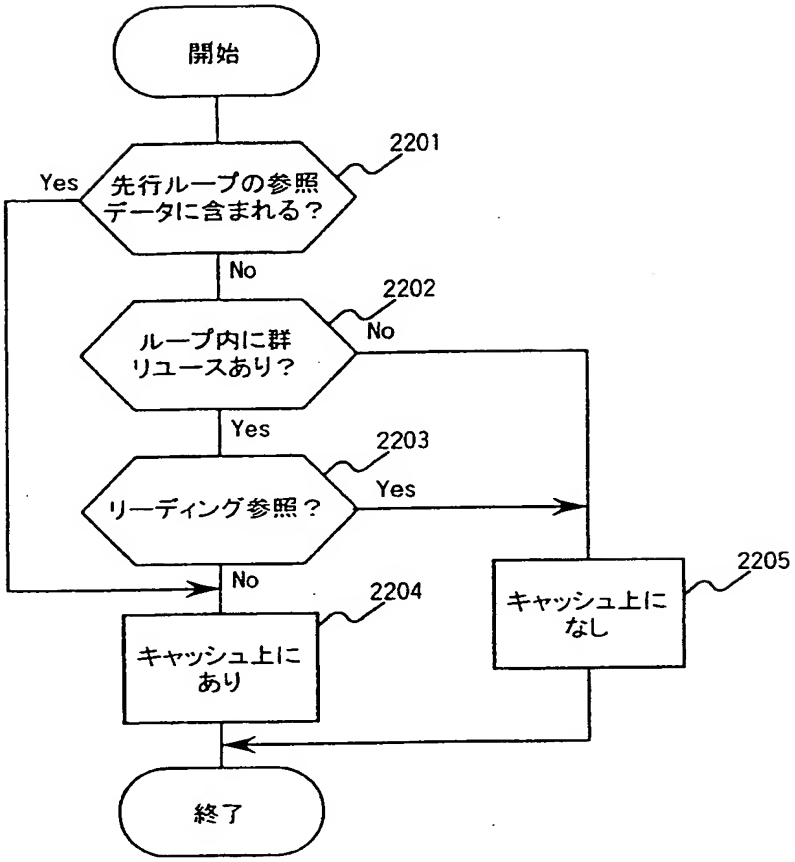
2004 2005 2006

群番号	リーディング参照	他の参照
1	a[i]	a[i+2]
2	d[i+100]	d[i]

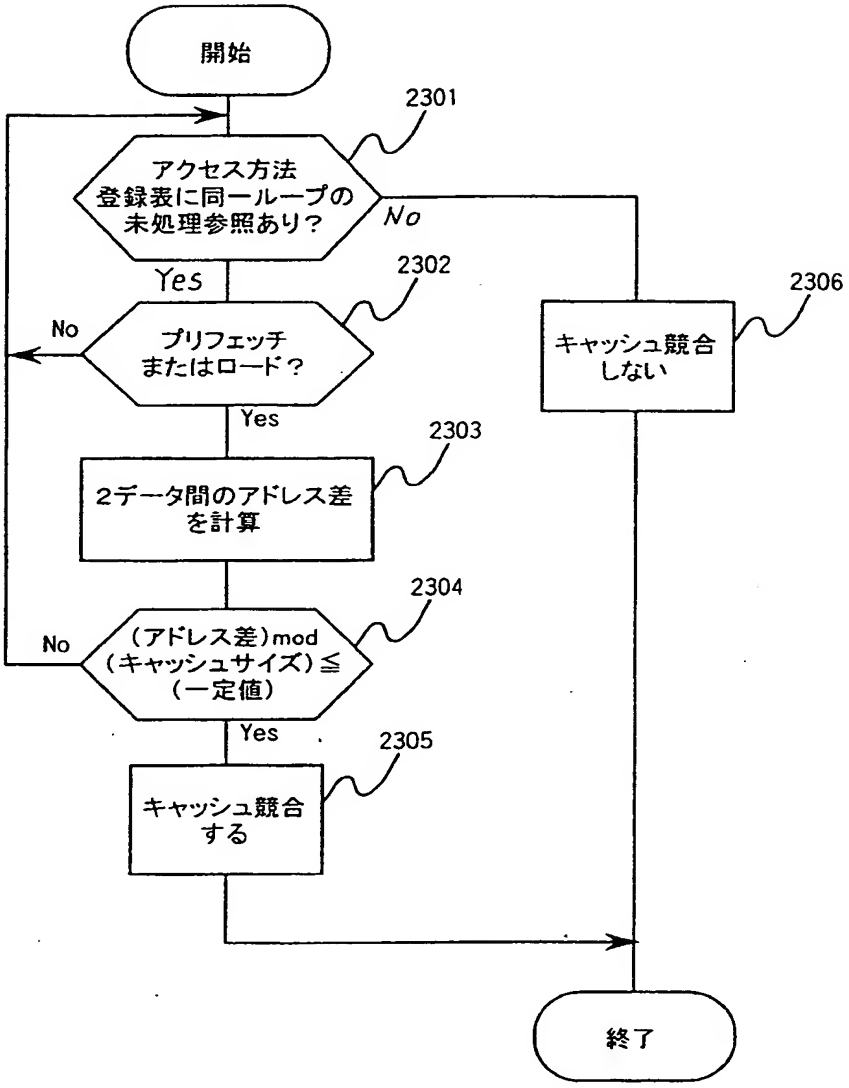
第２１図


開始
2101
すでにデータが
キャッシュ上にある？
Yes
No
2102
他のキャッシュデータと
キャッシュ競合？
Yes
No
2103
同一キャッシュライン
データを後で再参照？
No
Yes
2104
No
レジスタ資源の制約を
満たす？
Yes
2105
プリフェッチを
選択
2106
プリロードを
選択
2107
ロードを
選択
終了

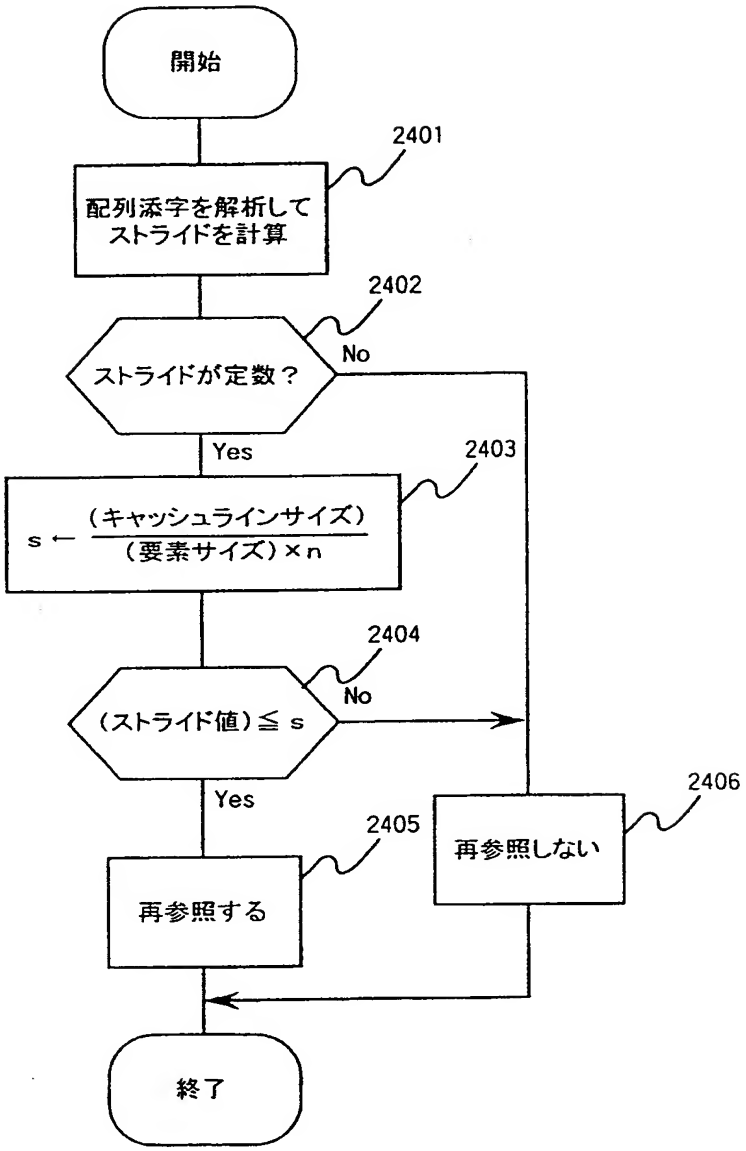
第２２図


開始
2201
先行ループの参照データに含まれる？
Yes
No
2202
ループ内に群リユースあり？
No
Yes
2203
リーディング参照？
Yes
No
2204
キャッシュ上にあり
2205
キャッシュ上になし
終了

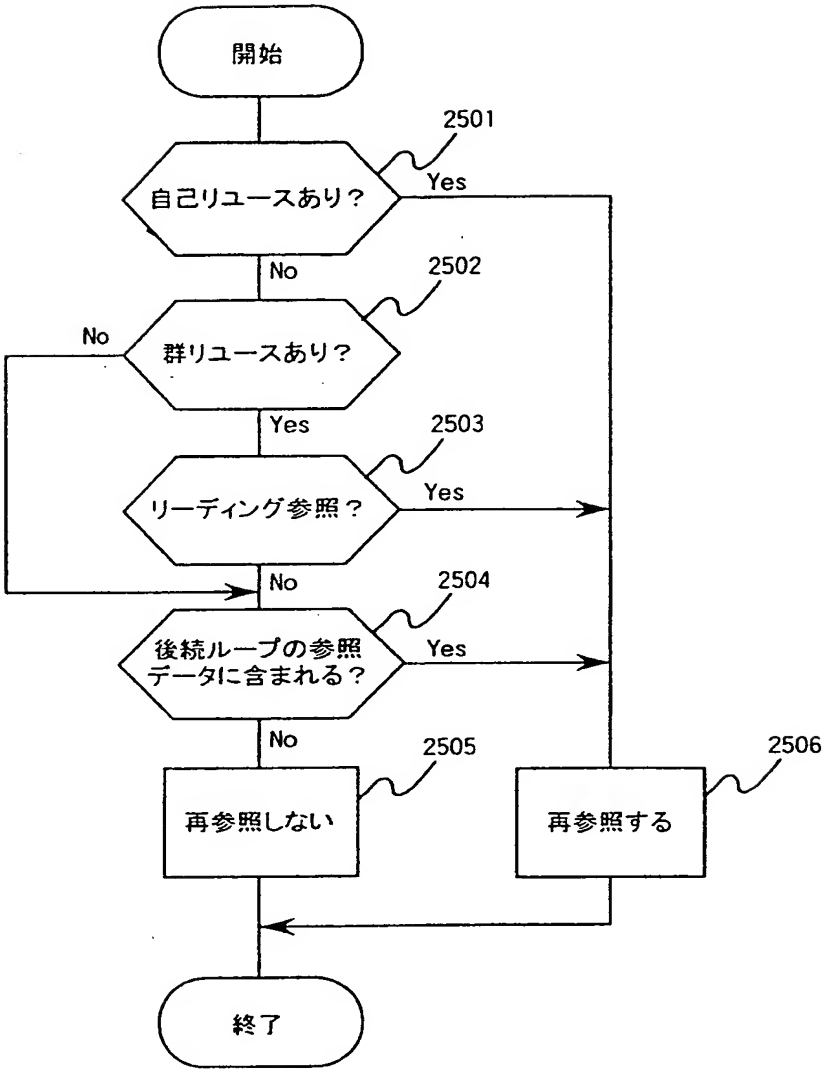
第２３図


開始
2301
アクセス方法
登録表に同一ループの
未処理参照あり？
No
Yes
2302
プリフェッチ
またはロード？
No
Yes
2303
2データ間のアドレス差
を計算
2304
(アドレス差)mod
(キャッシュサイズ)≦
(一定値)
No
Yes
2305
キャッシュ競合
する
2306
キャッシュ競合
しない
終了

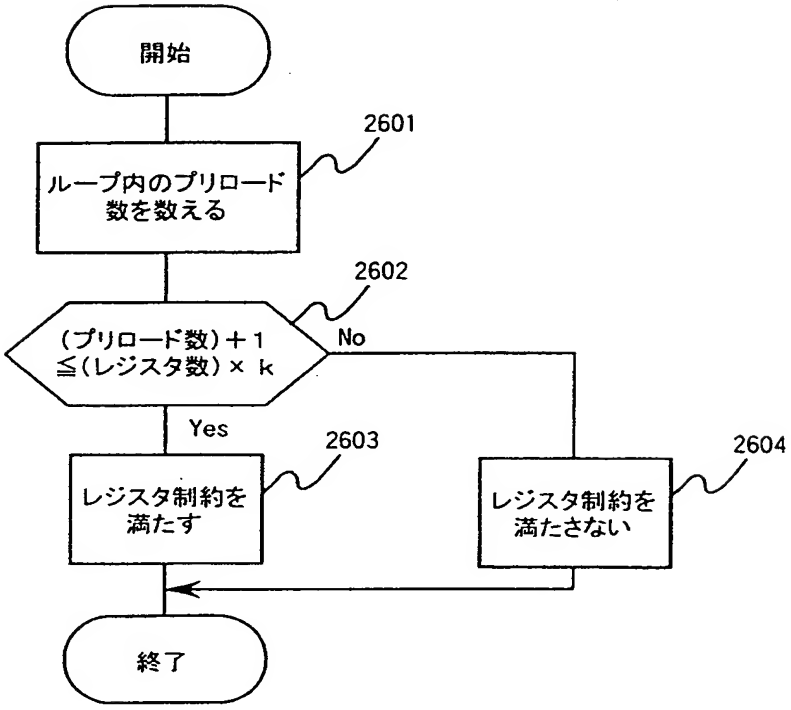
第２４図


開始
2401
配列添字を解析して
ストライドを計算
2402
ストライドが定数？
No
Yes
2403
s ← (キャッシュラインサイズ) / (要素サイズ)×n
2404
(ストライド値) ≦ s
No
Yes
2405
再参照する
2406
再参照しない
終了

第２５図


開始
2501
自己リユースあり？
Yes
No
2502
群リユースあり？
No
Yes
2503
リーディング参照？
Yes
No
2504
後続ループの参照データに含まれる？
Yes
No
2505
再参照しない
2506
再参照する
終了

第２６図


開始
2601
ループ内のプリロード数を数える
2602
(プリロード数)＋1 ≦(レジスタ数)× k
No
Yes
2603
レジスタ制約を満たす
2604
レジスタ制約を満たさない
終了

第27図

```
double  a[200],b[2000],c[100],d[200],s;
int     x[100];
void func()
{
    int     i;

    for (i = 0; i < 100; i++) {
        s += a[i] + a[i+2] + b[2*i] + c[x[i]]
             + d[i] + d[i+100] + b[2*i+1024];
    }
}
```

第２８図

```
double      v1, v2, v3, v4;
double      a[100], b[400], c[400];

void func()
{
    int     i;

    /* LOOP 1 */
    for (i = 0; i < 100; i++) {
        v1 += a[i];
    }

    /* LOOP 2 */
    for (i = 0; i < 100; i++) {
        v2 += a[i] + b[i*4] + c[i*4];
    }

    /* LOOP 3 */
    for (i = 0; i < 400; i++) {
        v3 += b[i];
    }

    /* LOOP 4 */
    for (i = 0; i < 400; i++) {
        v4 += c[i];
    }


}
```

INTERNATIONAL SEARCH REPORT

International application No.
PCT/JP97/04542

A. CLASSIFICATION OF SUBJECT MATTER
Int.Cl[6] G06F9/45

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
Int.Cl[6] G06F9/45, G06F12/08

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
Jitsuyo Shinan Koho 1971-1997 Toroku Jitsuyo Shinan Koho 1994-1997
Kokai Jitsuyo Shinan Koho 1971-1994

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	JP, 7-295882, A (Hitachi, Ltd.), November 10, 1995 (10. 11. 95) (Family: none)	1-9
A	Papers of Information Proceeding Society of Japan, Vol. 34, No. 5, April 1993, Hiroshi Nakamura et al., "Evalution of Pseudo-Vector Processor Using Register Window Method (in Japanese)" p.669-680	1-9
A	JP, 4-262458, A (NEC Corp.), September 17, 1992 (17. 09. 92) (Family: none)	1, 2

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier document but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
March 11, 1998 (11. 03. 98)	March 24, 1998 (24. 03. 98)
Name and mailing address of the ISA/ Japanese Patent Office	Authorized officer
Facsimile No.	Telephone No.

Form PCT/ISA/210 (second sheet) (July 1992)

国際調査報告

国際出願番号　PCT／JP97／04542

A.　発明の属する分野の分類（国際特許分類（IPC））
Int.Cl⁶ G06F9／45

B.　調査を行った分野

調査を行った最小限資料（国際特許分類（IPC））
Int.Cl⁶ G06F9／45
Int.Cl⁶ G06F12／08

最小限資料以外の資料で調査を行った分野に含まれるもの
日本国実用新案公報　1971－1997年
日本国公開実用新案公報　1971－1994年
日本国登録実用新案公報　1994－1997年

国際調査で使用した電子データベース（データベースの名称、調査に使用した用語）

C.　関連すると認められる文献

引用文献のカテゴリー＊	引用文献名　及び一部の箇所が関連するときは、その関連する箇所の表示	関連する請求の範囲の番号
A	JP，7－295882，A（株式会社日立製作所） 10.11月.1995（10.11.95）　（ファミリーなし）	1－9
A	情報処理学会論文誌，第34巻，第5号，4月.1993， 中村　宏 他「レジスタウィンドウ方式を用いた擬似ベクトルプロセッサの評価」　p.669－680	1－9
A	JP，4－262458，A（日本電気株式会社） 17.9月.1992（17.09.92）（ファミリーなし）	1，2

☐ C欄の続きにも文献が列挙されている。　☐ パテントファミリーに関する別紙を参照。

＊ 引用文献のカテゴリー
「A」特に関連のある文献ではなく、一般的技術水準を示すもの
「E」先行文献ではあるが、国際出願日以後に公表されたもの
「L」優先権主張に疑義を提起する文献又は他の文献の発行日若しくは他の特別な理由を確立するために引用する文献（理由を付す）
「O」口頭による開示、使用、展示等に言及する文献
「P」国際出願日前で、かつ優先権の主張の基礎となる出願の日の後に公表された文献
「T」国際出願日又は優先日後に公表された文献であって出願と矛盾するものではなく、発明の原理又は理論の理解のために引用するもの
「X」特に関連のある文献であって、当該文献のみで発明の新規性又は進歩性がないと考えられるもの
「Y」特に関連のある文献であって、当該文献と他の1以上の文献との、当業者にとって自明である組合せによって進歩性がないと考えられるもの
「&」同一パテントファミリー文献

国際調査を完了した日	国際調査報告の発送日
11．03．98	24.03.98

国際調査機関の名称及びあて先	特許庁審査官（権限のある職員）		
日本国特許庁（ISA／JP） 郵便番号100－8915 東京都千代田区霞が関三丁目4番3号	金田　利規	5B	9292
	電話番号　03－3581－1101　内線　3545		

様式PCT／ISA／210（第2ページ）（1992年7月）